MASTER EN SCIENCES INFORMATIQUES

Système de traitement de textes

Fourny, Françoise; Cheu, Jean-Marc

Award date:
1982

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR,

INSTITUT D'INFORMATIQUE

SYSTEME DE TRAITEMENT DE TEXTES.

Promoteur : Ph. Van Bastelaer.

Françoise Fourny.
Jean-Marc Cheu.

Mémoire présenté en vue
de l'obtention du grade de

LICENCIE ET MAITRE EN INFORMATIQUE

ANNEE ACADEMIQUE 1981 - 1982.

Table des matières

Avant propos.

Nous tenons à remercier pour l'aide apportée à ce mémoire, Monsieur Ph. Van Bastelaer, qui a dirigé ce mémoire et qui a permis, par ses conseils de mener à bien ce travail.

Nous tenons aussi à remercier Monsieur J-D. Nicoud et tous ses collaborateurs de nous avoir accueillis et aidés lors de notre travail en stage à l'EPFL.

Nous exprimons notre gratitude aux membres de l'institut, plus particulièrement J-P. Adans et R. Lesuisse, pour l'aide apportée par leurs conseils.

Introduction.

Ce mémoire aborde le sujet des systèmes de traitement de textes. A l'origine, les systèmes de traitement de textes étaient de simples outils d'introduction et d'édition de programmes. Mais, à l'heure actuelle, ils sont devenus plus puissants et sont reconnus comme une composante importante d'un système informatique. Ce progrès considérable a été permis grâce à trois choses : la diminution des coûts du matériel informatique, la disponibilité de nouveaux types de matériel (microprocesseur, équipements spécifiques au traitement de textes) et l'évolution des besoins, en particulier dans le domaine de la bureautique. Le traitement de textes est devenu une fin en lui-même et non plus un à-côté d'un système informatique.

Dans ce mémoire, le problème du traitement de textes est envisagé sous deux angles :

- Une présentation générale des différents types de systèmes existants.

- L'étude et la conception d'un système particulier.

Dans un premier chapitre, nous tentons de faire un tour d'horizon de quelques types de systèmes de traitement de textes représentatifs des tendances actuelles. Nous avons classifié ceux-ci selon leur orientation.

Les deux chapitres suivants présentent la démarche d'analyse conduisant à la réalisation du système qu'il nous a été demandé de réaliser dans le cadre de ce mémoire et en particulier de notre stage à l'Ecole Polytechnique Fédérale de Lausanne. Cet éditeur s'inspire de l'éditeur STAR de XEROX. Ces deux chapitres parlent de l'analyse fonctionnelle et de l'analyse organique du système. L'analyse fonctionnelle consiste en une démarche permettant d'établir un dossier qui détermine que réaliser afin de satisfaire aux exigences de l'utilisateur, tandis que l'analyse organique s'occupe de la façon dont seront réalisées ces exigences.

Nous parlons, dans un quatrième chapitre, des méthodes d'implémentation que nous avons utilisées. Ces méthodes comportent principalement deux volets : la méthode de programmation qui a été choisie pour réaliser cet éditeur et la méthode de test qui a été utilisée.

Le cinquième chapitre constitue le manuel d'utilisation de l'éditeur réalisé. Nous y expliquons comment utiliser l'outil de traitement de textes qui a été implementé.

Dans le dernier chapitre de ce mémoire, nous parlons du matériel pour lequel cet éditeur a été écrit et de la portabilité de ce système. L'application a été réalisée de façon a être implémentée avec peu de modifications sur deux matériels différents.

Pour terminer, nous tirons un certain nombre de conclusions. Nous expliquons ce qui a pu être réalisé et donnons les raisons qui furent la cause de ce que tout ne put être implémenté. Ensuite, nous suggérons une liste d'améliorations à apporter à ce qui a été réalisé.

Chapitre 1

QUELQUES SYSTEMES DE TRAITEMENTS DE TEXTES ET QUELQUES EDITEURS

Chapitre 1: Quelques systèmes de traitement de texte et quelques éditeurs

(Jean-Marc Cheu)

1. Introduction

1.1. Vue générale

On peut communément distinguer plusieurs systèmes de traitement de texte. Nous proposons de répartir ces systèmes en trois catégories:

- les systèmes de traitement de mots (S.T.M.) ou "word processing"
- les systèmes de traitement de documents (S.T.D) ou "document processing"
- les systèmes de traitement de programmes (S.T.P) ou "program processing"

Un système de traitement de mots a comme but la création, la maintenance, le formatage et l'impression de textes. On entend ici par "texte" une suite de caractères limités aux caractères alphanumériques et aux signes de ponctuation. Pour arriver à ses fins, il dispose de trois fonctions spécifiques qui sont :

- l'édition de texte qui comprend la création, la maintenance et l'archivage de ces textes
- le formatage du texte qui modifie le format de celui-ci c'est-à-dire sa mise en page ainsi que le type de caractère (police)
- les utilitaires de manipulation globale des textes tels que l'impression, la copie, la suppression, etc...

Un système de traitement de documents a les mêmes buts qu'un système de traitement de textes mais appliqués à des documents. La notion de "document" englobe des objets très différents de par leur nature tels que du texte, des équations, des tableaux, des diagrammes, des photographies et des documents graphiques. En général, on y trouvera donc tout ce qu'on peut voir sur une page imprimée.

Un système de traitement de programmes, quant à lui, a pour but la création, la maintenance et l'archivage de programmes d'ordinateurs qu'ils soient ou non représentés sous une forme

textuelle. Certains de ces systèmes traitent le contenu sémantique du "texte" programme; par exemple, ils réalisent un contrôle syntaxique ou encore ils assurent une présentation basée sur les règles du langage, etc...

Les principales caractéristiques sur lesquelles pourrait se baser une évaluation de ces systèmes sont:

- l'existence de commandes considérées comme essentielles à l'édition de texte telles que la recherche de la première apparition d'un objet spécifié, la substitution d'un objet par un autre pour chacune de ses occurrences, la copie d'une phrase, d'un paragraphe, la reconnaissance de structures (mots, phrases, alinéa, paragraphes, ...), etc...

- un interface utilisateur concis et cohérent, facile à utiliser et à apprendre

- la possibilité de présenter à l'écran le document dans un format aussi proche que possible de celui sous lequel il sera imprimé.

- un temps de réponse rapide

- un accès partagé et contrôlé de mêmes fichiers par plusieurs utilisateurs simultanés

Nous présentons ci-dessous un bref aperçu historique de l'évolution de ces systèmes.

1.2. Historique

Initialement, c'est-à-dire aux environs des années 50, les éditeurs furent utilisés pour créer et modifier des programmes et les données de ces programmes. Ils forment le groupe connu des éditeurs de programme. L'unité de base de l'information est alors le bloc de 80 caractères correspondant au contenu d'une carte perforée.

Une première méthode pour l'édition de programmes fut l'éditeur batch. Le programme initial étant sur fichier, on introduisait une série de cartes de correction. Ces corrections étaient effectuées par un programme utilitaire appelé l'éditeur batch.

La seconde méthode fut, vers l'année 1965, d'effectuer, à partir d'une console télétype, des corrections sous contrôle d'un programme d'édition.

Au début des années 70, vinrent les éditeurs permettant de traiter des lignes de longueur variable. L'unité d'opération reste cependant la ligne.

L'évolution majeure par rapport à la vision ligne par ligne fut l'apparition des éditeurs orientés écran vers l'année 75. Ceux-ci permettent non seulement de visualiser plusieurs

lignes en même temps mais en plus ils offrent la possibilité de déplacer un curseur dans cet écran. A partir de là, l'édition de programmes est beaucoup plus facile et la notion d'édition de texte apparait.

A partir des années 60, en parallèle avec les éditeurs de lignes, des éditeurs graphiques furent conçus pour permettre à l'utilisateur de manipuler des objets graphiques.

Depuis 5 ans, l'évolution s'accentue de plus en plus. En effet, le développement des terminaux à mémoire locale a permis de faire de l'édition locale ce qui va dans le sens d'une plus grande autonomie pour les terminaux et d'une diminution de travail pour l'ordinateur central. Dans le milieu des années 70, la disponibilité et la diminution du coût des mini- et micro-ordinateurs encouragent l'expansion de l'industrie du traitement de texte. De plus, l'édition de texte et l'édition graphique sont regroupées de manière à aborder le traitement de documents. Pour accompagner ces innovations, les possibilités graphiques des terminaux sont augmentées ; des éditeurs à fenêtres multiples sont créés et de nouveaux moyens de pointage sur l'écran sont construits (la souris, le palpeur, le crayon lumineux, le manche à balai, etc...).

Les recherches actuelles dans le domaine de l'édition évoluent dans trois directions :

- la création d'interfaces plus cohérents, agréables et simples pour un utilisateur.

- la création d'éditeurs de structure reconnaissant au niveau des commandes de l'éditeur, les différents niveaux structurels du texte tels que le mot, l'alinéa, le paragraphe, etc...

- la création d'éditeurs "fidèles" ou selon la terminologie américaine que nous emploierons ici "what you see is what you get" qui permettent des opérations d'édition sur un texte à l'écran qui est une représentation exacte d'une page imprimée. Ceci est rendu possible par la haute définition tant des écrans que des imprimantes (1000 points par ligne pour les imprimantes et 800 x 1000 points pour un écran de 11 pouces).

Nous allons maintenant essayer de classifier ces différents éditeurs selon certains critères.

1.3. Critères de classification

Cette classification prend en compte 4 caractéristiques que nous considérons comme importantes :

- le but de l'éditeur
- l'affichage
- l'unité
- l'interface

Parmi ces caractéristiques, le but de l'éditeur et ce qui en découle c'est-à-dire son mode d'affichage constituent la manière la plus commune de classifier les éditeurs. En effet, il existe généralement une relation étroite entre le but de l'éditeur et le mode d'affichage de cet éditeur. A chaque catégorie d'éditeurs correspond, la plupart du temps, un mode d'affichage qui lui est propre. Nous avons laissé de côté les éditeurs spécifiquement et exclusivement orientés graphique car nous nous en sommes tenus autant que possible à tout ce qui concernait le traitement de textes, de programmes et de documents.

1.3.1. Le but de l'éditeur

Parmi les buts de ces éditeurs, on peut distinguer trois grandes catégories :

- les éditeurs de programmes
- les éditeurs de textes
- les éditeurs de documents.

Depuis l'avènement de la carte perforée, les éditeurs de programmes simulent souvent la structure d'une carte. A cause de cela, la plupart des éditeurs ont un mode d'affichage orienté lignes. Bien entendu, ces éditeurs peuvent aussi traiter du texte bien qu'ils ne soient pas spécifiquement prévus pour ce travail.

Les éditeurs de textes, comme l'indique leur nom, traitent du texte. De ce fait, leur mode d'affichage est généralement orienté écran. Dans ces éditeurs, nous pouvons distinguer, parmi d'autres éditeurs spécialisés, les éditeurs orientés structure. Ces derniers tiennent compte, pour les commandes de l'éditeur, de l'élément structurel pointé.

Les éditeurs de documents, grâce à de puissants moyens graphiques et des moyens de formatage permettent de traiter des documents complexes composés de textes et de graphiques. Selon cette puissance graphique et ces formateurs, nous pouvons distinguer des éditeurs de documents classiques et des éditeurs de documents "what you see is what you get". Ces derniers tentent de rendre sur l'écran du terminal la vue la plus fidèle possible de ce que sera le document une fois imprimé. Ces éditeurs ont aussi

un mode d'affichage orienté écran mais cette fois, le texte sera formaté et directement visible comme tel sur l'écran.

1.3.2. Le mode d'affichage

Le mode d'affichage est le moyen par lequel nous pouvons voir les programmes, les textes et les documents. Deux grands modes d'affichage peuvent être distingués :

- orienté lignes
- orienté écran

Le mode d'affichage orienté lignes signifie que nous allons voir à l'écran une ligne ou un ensemble de lignes. L'utilisateur ne peut pas déplacer le curseur dans une ligne et dans le texte comme il le veut. Il reste donc toujours dans le mode de commandes de l'éditeur. Toutes les commandes se réfèrent à la notion de "ligne".

Le mode d'affichage orienté écran, quant à lui, signifie que ce qui est vu à l'écran est un "texte". Cela signifie que l'utilisateur déplace le curseur où il le veut dans l'écran et tout ce qui est tapé est introduit à la position du curseur et est considéré comme du texte. L'utilisateur est donc généralement en mode d'insertion.

1.3.3. L'unité

L'unité est l'objet que vont traiter les commandes de ces éditeurs. Cet objet peut être de différentes sortes:

- la ligne
- le symbole
- le caractère
- l'élément de structure

L'unité "ligne" est une des plus connues. Les commandes des éditeurs vont traiter des lignes identifiées par des numéros de lignes. Le plus souvent, ce sont ces numéros qui vont apparaître dans les commandes. Ces lignes peuvent être de longueur fixe ou de longueur variable.

L'unité "symbole" va permettre de traiter le contenu des lignes c'est-à-dire des mots. Ceux-ci sont des suites de caractères alphanumériques précédées et suivies de blancs.

L'unité "caractère" ,quant à elle, va permettre de traiter le contenu des mots c'est-à-dire les caractères.

L'unité "élément de structure" peut être un caractère, un mot, un alinéa, un paragraphe ou un chapitre. Cette unité un peu spéciale puisqu'elle peut s'accroître du point de vue de son contenu est directement liée à la

manière interne dont le texte est structuré. Cette structure est souvent une structure d'"arbre" dont les éléments sont les "noeuds". Ces noeuds sont répartis entre les différentes "branches" de l'"arbre". Cette structure interne n'est pas visible de l'extérieur car l'utilisateur, quand il déplace un mot, n'a pas l'impression de changer un "noeud" dans un "arbre" ni de sauter de "branche" en "branche".

1.3.4. L'interface

L'interface représente le moyen dont dispose l'utilisateur pour donner des ordres à ces éditeurs. Ce moyen peut être fort différent selon les éditeurs. On peut ainsi distinguer :

- les mots à taper en clair
- les touches de fonction
- les touches de contrôle
- les touches du clavier
- les touches du curseur

L'interface "mots à taper en clair" signifie que l'utilisateur doit écrire en toute lettre le nom de la commande.

L'interface "touches de fonction" est constitué par des touches qui, à elles seules, représentent une commande complète. Par exemple, nous pourrions avoir une touche "annule", une touche "exécute" une touche "imprime", etc...(cfr éditeur BEEZY).

L'interface "touches de contrôle" consiste en un certain nombre de touches de contrôle qui servent surtout en combinaison avec une lettre alphanumérique du clavier.

exemple : control / v signifie "passer à l'écran suivant"

Elles permettent de donner tous les ordres de base d'un éditeur mais leur problème est leur signification car il faut se rappeler ce que veut dire chaque touche.

L'interface "touches du clavier" représente les touches alphanumériques du clavier. Suivant le mode dans lequel on se trouve, ces touches ont une signification particulière et elles permettent d'exécuter des ordres différents selon les modes. Ces touches sont en général l'initiale de la commande à exécuter.

L'interface "touches du curseur" sert surtout au déplacement du curseur mais ces touches peuvent aussi servir de facteur répétitif, c'est-à-dire que dans certains cas, elles continuent l'effet de la commande précédente (cfr

éditeur PASCAL UCSD).

1.4. L'éditeur lui-même

L'éditeur se particularise par la forme de ses commandes et par les fonctions qu'il fournit en plus ou en moins par rapport à celles de base de sa catégorie.

Pour chaque éditeur, nous donnons la forme de ces commandes. En effet, si le but de ces commandes est toujours le même, la forme diffère souvent selon le type d'éditeur.

Nous donnons ensuite les diverses particularités que comporte chaque éditeur.

1.5. La revue des éditeurs

Nous allons maintenant passer en revue un certain nombre d'éditeurs classés selon leur but et leur mode d'affichage.

Les différentes catégories d'éditeurs considérés sont donc :

- les éditeurs orientés lignes
- les éditeurs orientés écran
- les éditeurs orientés structure
- les éditeurs orientés documents
- les éditeurs orientés "what you see is what you get"
- les éditeurs divers

Nous avons choisi ces éditeurs non parce qu'ils sont les meilleurs de chaque classe mais parce qu'ils en sont une bonne représentation. A la fin de cette classification, nous appelons "éditeurs divers" quelques éditeurs qui, pour une raison ou pour une autre, n'entrent pas vraiment dans les différentes catégories qui précèdent.

A la fin de cette description malgré tout assez superficielle vu le temps et les moyens dont nous disposions, nous décrirons le système Smalltalk dont la philosophie est radicalement différente de celle de tous les autres éditeurs.

En guise de conclusion, nous donnerons enfin deux tableaux synoptiques regroupant tous les éditeurs présentés ainsi que leurs principales caractéristiques et une réflexion sur l'orientation des éditeurs.

2. Les éditeurs orientés lignes

2.1. Préliminaires.

Les éditeurs orientés lignes sont les premiers éditeurs à avoir été commercialisés sur les systèmes d'ordinateurs. Ils peuvent être soit à lignes de longueur fixe soit à lignes de longueur variable.

Rappelons les caractéristiques d'un éditeur orienté lignes : L'utilisateur ne peut pas déplacer le curseur dans une ligne et dans le texte comme il le veut. Il reste donc toujours dans le mode de commandes de l'éditeur.

Les fonctions de base d'un éditeur orienté lignes sont les suivantes :

- recherche d'une suite de caractères
- changement d'une suite de caractères
- numérotation des lignes
- insertion de lignes
- destruction de lignes
- transfert de lignes
- copie de lignes
- impression de lignes à l'écran
- sauvetage des modifications apportées
- sortie de l'éditeur

2.2. L'éditeur CMS d'IBM. [1]

but de l'éditeur	: programmes (texte)
affichage	: ligne
unité	: ligne de longueur fixe
interface	: mots à taper en clair

Forme des commandes.

Sur un court exemple, on peut voir comment fonctionne cet éditeur et quelle est la forme des commandes.

exemple :	Instruction	Résultat à l'écran
	find add:	add: procedure
	change/add/substract	substract: procedure

La suite de caractères "add:" est localisée en utilisant la

commande "find". Le pointeur de la ligne courante, une entitè qui marque la fenêtre courante, pointe à la ligne "add:procèdure;". Cette ligne est écrite sur l'écran. La commande "change/add/substract" affecte seulement le contenu de la fenêtre: la première apparition de "add" est remplacée par "substract". La forme des commandes est donc le nom de la commande suivi de l'objet de cette commande.

Particularitès.

L'éditeur assure le changement automatique de minuscule en majuscule.

Si la longueur maximum de ligne de 132 caractères est dépassée, la ligne sera tronquée.

2.3. L'éditeur ED de UNIX. [2]

but de l'éditeur	: programmes (texte)
affichage	: ligne
unitè	: ligne de longueur variable
interface	: mots à taper en clair

Forme des commandes.

La forme gènèrale d'une commande de ED est le nom de la commande, parfois prècèdè par un ou deux numèros de ligne et parfois suivi par un ou plusieurs paramètres. Une seule commande est permise par ligne mais la commande p(rint) peut en suivre quelques autres.

Particularitès.

Il existe une commande qui permet d'imprimer l'environnement (dans la plupart des cas, les 10 lignes prècèdentes et les 9 lignes suivantes) de la ligne courante.

Une caractèristique intèressante de ED est la possibilitè de travailler en mode visuel. Cela signifie que l'utilisateur a constamment à l'écran la vision de l'environnement de la ligne courante. Ceci permet de simuler à peu près un éditeur orientè écran à part le fait que l'on ne sait pas se dèplacer à l'intèrieur d'une ligne.

3. Les éditeurs orientés écran

3.1. Préliminaires

Rappelons qu'un éditeur orienté écran est caractérisé par le fait qu'il permet la manipulation d'un écran considédé comme un tout et qu'il se base essentiellement sur l'emploi d'un pointeur permettant à tout moment la référence à un point quelconque de cet écran.

Les fonctions de base de tout éditeur orienté écran sont:

- les fonctions de base d'un éditeur orienté lignes
- la possibilité de déplacer le curseur d'un endroit quelconque de l'écran à tout autre endroit.
- la possibilité de passer à l'écran suivant, à l'écran précédent, à l'écran initial et à l'écran final.

3.2. L'éditeur BB de BROWN. [1]

but de l'éditeur	: texte(programme)
affichage	: écran
unité	: caractère
interface	: touche de fonction/ touche de contrôle.

Forme des commandes.

Les commandes doivent être entrées soit par des touches de fonction, des caractères de contrôle, des séquences d'"escape" ou soit par l'utilisation d'une ligne spéciale de commande au sommet de l'écran.

Particularités.

Il existe une ligne de rappel des commandes en haut de l'écran.

L'éditeur reconnait différentes unités de travail (mots, lignes, paragraphes) et il offre la possibilité d'actions sur ces unités.

L'éditeur BB maintient un historique des touches poussées et cela pour chaque touche dans une session d'édition. Dans l'éventualité d'un arrêt accidentel du système, des fichiers de backup crées automatiquement peuvent être exécutés avec l'historique des touches poussées pour retrouver toutes les opérations sauf la dernière.

L'éditeur BB examine la partie extension du nom de fichier du document courant et charge une table interne avec des informations qui varient avec l'extension. Ceci permet à BB de faire de l'indentation automatique pour des langages de programmations variés et pour reconnaitre des entités

structurelles telles que des paragraphes dans les documents.

L'éditeur BB supporte des fenêtres multiples d'édition. La fenêtre de l'écran peut être bougée vers la gauche et vers la droite tandis que des fenêtres multiples supportent facilement l'édition simultanée de plusieurs documents.

3.3. L'éditeur Z de YALE. [1]

but de l'éditeur	: texte (programme)
affichage	: écran
unité	: caractère
interface	: touche de fonction/ touche de contrôle.

Forme des commandes.

Les commandes de Z sont tapées en utilisant des caractères de contrôle couplés avec les touches du curseur.

Particularités.

Pour se déplacer, Z emploie un curseur. Ceci est facilité par le fait qu'il se rappelle les 7 dernières positions de la fenêtre. Cela permet à l'utilisateur de revoir ses précédents contextes pendant que la fenêtre courante ne change pas.

Une autre commande permet à l'utilisateur de mettre une marque dans le fichier pour pouvoir y retourner plus tard.

L'éditeur Z a une commande de justification pour formater un ou plusieurs paragraphes à la fois.

3.4. L'éditeur EDIS. [10]

but de l'éditeur:	programme
affichage:	écran
unité:	symbole
interface:	touches du clavier

Forme des commandes.

Les commandes doivent être entrées en utilisant une ligne de rappel des commandes qui se trouvent en haut de l'écran. L'utilisateur sélectionne une commande en tapant l'initiale de cette commande.

Particularités.

Il existe une ligne de rappel des commandes.

Lors de l'insertion de plusieurs lignes, la marge reste à la valeur de la ligne précédente (sauf indication contraire).

Les symboles particuliers du langage PASCAL peuvent être introduits à l'aide d'une touche du clavier (en majuscule). Si les majuscules sont nécessaires dans leur acception courante, on peut supprimer cette interprétation. La correspondance est la suivante :

A	array	J	integer	S	record
B	begin	K	case	T	type
C	const	L	else	U	until
D	do	M	string	V	var
E	end	N	then	W	while
F	for	O	of	X	*)
G	program	P	procedure	Y	with
H	char	Q	function	Z	(*
I	if	R	repeat	$	(*$L LISTING.TEXT*)

Les déplacements du curseur peuvent être synchronisés avec les symboles du programme : le curseur sautera au début du nombre, identificateur, chaine ou commentaire suivant.

La forme des instructions, des déclarations, des blocs et modules peut être vérifiées conformément au langage PASCAL UCSD par une commande d'analyse syntaxique et à un faible niveau, d'analyse sémantique d'un morceau de texte.

3.5. L'éditeur PASCAL UCSD

but de l'éditeur:	texte et programme
affichage:	écran
unité:	caractère
interface:	touches du clavier

Forme des commandes.

Pour les commandes de l'éditeur, l'utilisateur a sous les yeux une ligne de commandes qui rappelle le nom des différentes commandes disponibles. Une fois que l'utilisateur a sélectionné une commande en tapant l'initiale de cette commande, cette ligne de rappel va contenir les diverses possibilités attachées à la commande sélectionnée.

Particularités.

Il existe une ligne de rappel des commandes en haut de l'écran.

L'éditeur PASCAL UCSD dispose d'une fonction d'indentation automatique.

Une fonction intéressante est l'ajustement d'une ligne. On peut ainsi déplacer horizontalement toute une ligne. Mais en plus, cet ajustement a un facteur répétitif. Par exemple, si on déplace une ligne de 3 caractères vers la gauche et que l'on veuille faire la même chose pour les 5 lignes suivantes, il suffit de se déplacer verticalement avec les touches de déplacement du curseur et l'ajustement se fera automatiquement.

Avant de terminer l'exécution d'une commande, ce qui se fait dans tous les cas par la même touche, l'utilisateur peut taper une touche spéciale qui annulera l'effet de cette commande.

4. Les éditeurs orientés structure.

4.1. Préliminaires

Les éditeurs orientés structure ont été "redécouverts" comme une méthode alternative d'édition standard orientée texte. Comme la plupart des applications ont une certaine structure interne (les documents écrits sont divisés en chapitres, sections, paragraphes, etc...), la philosophie des éditeurs orientés structure est d'exploiter cette division pour simplifier l'édition. La représentation la plus commune est la structure d'arbre hiérarchique. Le sous-arbre est une partie d'un arbre qui a lui-même une structure d'arbre.

Les fonctions de base des éditeurs orientés structure sont :

- descendre d'un niveau dans l'arbre
- monter d'un niveau dans l'arbre
- aller à la tête d'un sous-arbre
- aller à la fin d'un sous-arbre
- insérer un nouveau noeud (avec une collection de données vide) dans un endroit spécifié
- détruire un sous-arbre (noeud et données)
- copier un sous-arbre dans un endroit spécifié
- bouger un sous-arbre vers un endroit spécifié
- couper un noeud et ses données en deux
- mélanger deux noeuds et leurs données
- insérer un niveau intermédiaire dans l'arbre
- détruire un niveau intermédiaire de l'arbre
- permuter les noeuds sur un niveau de l'arbre

La formalisation des objets consiste à définir leur structure. Cette structure doit correspondre de près à la sémantique. Cela veut dire que les différentes structures choisies devront correspondre à des morceaux de texte ayant une certaine unité au point de vue sens. La première étape est de définir les noms des différentes structures choisies. Les noms suivants ont été choisis: le document, le paragraphe, l'alinéa, la liste, l'élément de liste, la phrase, le titre et le mot. Les noms de ces éléments de structure seront leur type de structure.

On définit formellement ces types de structure comme suit:

- le texte au point de vue formel est appelé un document.
- un document est défini comme une suite de paragraphes et/ou d'alinéas.
- un paragraphe est défini comme une suite de paragraphes et/ou d'alinéas précédée d'un titre. C'est donc une structure récursive.
- un alinéa est défini comme une suite de phrases.
- une phrase est formée de mots.
- un titre est formé de mots.
- un mot est la plus petite unité sémantique du texte.

La composition physique du mot, sa représentation, est celle d'une suite de caractères. Les signes de ponctuation, les parenthèses servent à définir des unités sémantiques comprises entre la phrase et le mot. On les a donc défini comme des mots. La structure ainsi définie est représentable par un arbre dont les noeuds sont les différents éléments de structure.

4.2. L'éditeur THOT. [6]

but de l'éditeur	: texte
affichage	: écran
unité	: elément de structure
interface	: touches de fonction

Forme des commandes.

La forme des commandes répond à la règle "définir / opérer".

La formalisation a débuté par le choix de ne définir que des opérations à une opérande. Le cas de la destruction et de l'insertion ne pose aucun problème. Par contre, le déplacement de texte impose de diviser l'opération en deux. On a décidé qu'un déplacement se fera comme suit : d'abord détruire la structure à déplacer puis la "ressusciter" au bon endroit. Cela consiste en fait à rendre la destruction réversible. Ensuite, chaque opération a été divisée en deux phases: une phase d'accès à la structure à opérer et une phase d'opération proprement dite. La première phase est dite "mouvement" dans le texte, la seconde est dite "opération" dans le texte.

Les primitives de "mouvement" sont au nombre de quatre :

_ "descendre" : va d'un noeud à son premier descendant du niveau suivant.

_ "monter" : va d'un noeud à son ascendant du niveau précédent.

_ "aller à droite" : va au noeud suivant de même niveau.

_ "aller à gauche" : va au noeud précédent de même niveau.

Les primitives d'"opération" sont aussi au nombre de quatre :

_ "insérer en dessous" : insère en dessous du noeud courant.

_ "insérer à gauche" : insère à gauche du noeud courant.

_ "détruire" : détruit le noeud courant.

_ "ressusciter" : rappelle le dernier noeud détruit et l'insère à gauche du noeud courant.

Particularités.

Il existe une fonction de formatage automatique d'un texte.

A chaque type de structure, on associe une représentation physique, modifiable à tout instant par l'utilisateur. Cette représentation est appellée cadre. Ce cadre définit d'une part les marges à gauche et à droite relatives, c'est-à-dire que si un noeud dans l'arbre possède un type donné, la marge de la structure correspondant à ce noeud est fixée, en accord avec le cadre du type de structure. Cela permet de traiter correctement la marge des structures récursives.

D'autre part, le cadre contient deux autres éléments :

- l'entête qui est l'indicatif de début de structure.
- la finale qui marque la fin de la structure.

5. Les éditeurs orientés documents

5.1. Préliminaires

Les éditeurs orientés documents sont des éditeurs spécialisés dans l'édition et la préparation de documents complexes qui regroupent du texte et des figures.

Les fonctions de base d'un éditeur orienté documents sont celles d'un éditeur orienté structure avec toutefois des éléments de structure supplémentaires :

- le tableau
- la figure

En plus de ces fonctions, il y a des fonctions spécifiques aux nouveaux éléments de structure.

5.2. L'éditeur de document PEN de YALE. [1]

but de l'éditeur:	documents
affichage:	écran
unité:	élément de structure
interface:	non-spécifié

Forme des commandes.

La forme des commandes n'était pas spécifiée dans la documentation qui était à notre diposition.

Particularités.

L'existence de fenêtres multiples d'édition.

La possibilité d'écrire un texte espacé proportionnellement avec des fontes diverses.

Comme dans les éditeurs orienté structure, la représentation d'un document dans PEN est l'arbre, avec des noeuds contenant des entités structurelles telles que des chapitres et des paragraphes. Un noeud est strictement défini par un "template" qui contient :

- des informations sur la structure du descendant de ce noeud
- des informations appartenant aux paramètres de ce noeud
- des informations appartenant au formatage de ce noeud
- des paires message/fonction qui permettent au noeud de réagir à la réception de messages particuliers
- des informations appartenant au cas par défaut de ce noeud.

6. Les éditeurs orientés "what you see is what you get"

6.1. Préliminaires

Les éditeurs orientés "what you see is what you get" représentent le dernier cri en matière d'éditeurs de texte. Ils couplent la facilité de se déplacer dans l'écran (comme les éditeurs orientés écran) et la possibilité de voir le texte tel qu'il apparaîtra réellement sur la feuille de papier. Tout cela se fait grâce à de puissants moyens graphiques et à des moyens de formatage.

Les fonctions de base d'un éditeur orienté "what you see is what you get" sont :

- les fonctions d'un éditeur orienté documents
- le formatage interactif

6.2. L'éditeur BRAVO de XEROX. [1] [12]

but de l'éditeur:	texte
affichage:	écran
unité:	élément de structure
interface:	touche de contrôle / souris

Forme des commandes.

L'utilisateur communique la plupart de ces ordres pour BRAVO au moyen des commandes du clavier. La plupart sont de simples lettres ou des séquences CTRL/lettre. Le reste des commandes est donné au moyen de la souris.

BRAVO est construit sur les concepts de sélection et de commande. Une sélection est simplement l'unité dans le texte avec laquelle quelqu'un veut opérer et la commande est ce que ce quelqu'un veut faire avec l' unité choisie. L'utilisateur doit toujours spécifier la sélection avant la commande.

Particularités.

Couplé avec les systèmes d'édition de dessins DRAW et MARKUP, BRAVO permet la création et la révision d'un document contenant à la fois du texte et des dessins tout en maintenant la philosophie du "what you see is what you get" pour le texte.

BRAVO a une commande pour annuler l'effet d'une commande d'un seul niveau. Simplement en tapant U l'utilisateur peut revenir dans la situation dans laquelle il se trouvait avant sa dernière commande.

L'unité pour la spécification des attributs de formatage de texte est la "view". Chaque caractère dans le document est associé à une "view" particulière. La "view" de chaque caractère peut être affichée en sélectionnant le caractère et en tapant L?

. La "view" spécifie un large assortiment des attributs du type : le style de police, la taille du caractère, le soulignage, le surlignage, le centrage, la justification, l'indentation et l'interligne.

Une autre caractéristique, qui est permise par la grande puissance graphique de l'écran, est la vision proportionnelle des caractères.

L'écran de BRAVO est divisé en plusieurs fenêtres. Le système de fenêtres contient des informations concernant ce que l'utilisateur vient juste de faire et ce qu'il peut maintenant faire. La fenêtre document contient le texte réel du document.

6.3. L'éditeur STAR de XEROX. [4]

but de l'éditeur:	document
affichage:	écran
unité:	élément de structure
interface:	touche de contrôle avec une souris.

Forme des commandes.

STAR suit la convention d'édition "définir/opérer". L'utilisateur définit l'objet à opérer en le pointant et en poussant le bouton gauche de la souris. Pousser une première fois définira le caractère sous le curseur, une deuxième le mot, une troisième la phrase, une quatrième fois le paragraphe.

En plus des touches normales du clavier, STAR affiche sur l'écran un petit nombre de commandes. L'utilisateur peut exécuter ces commandes en les pointant avec la souris et en pressant le bouton de sélection sur la souris.

Particularités.

La première chose à comprendre sur STAR est qu'il n'est pas un produit qui fut conçu et spécifié par des stratèges du marché. Il résulte plutôt des experiences de chercheurs qui développent des outils pour leur propre usage et pour explorer de nouvelles idées sur l'utilisation d'ordinateurs.

Beaucoup d'efforts de conception de système commencent par des spécifications matérielles et sont suivis par une série de spécifications fonctionnelles pour le logiciel. Le projet STAR commenca d'une autre manière : La première tâche fut de définir un modèle conceptuel de la façon dont l'utilisateur serait relié au système. Le matériel et le logiciel découlent de cela.

L'équipe de chercheurs se dirigea dans la direction d'un utilisateur novice et la philosophie du système s'en ressentit :

- 1. VOIR ET POINTER plutôt que SE SOUVENIR ET TAPER.
  L'utilisateur ne doit pas se souvenir des commandes. Il doit être capable d'exécuter une commande simplement en la pointant.

- 2. UNIFORMITE DES COMMANDES AU TRAVERS DES DOMAINES.
  Une fois que l'utilisateur a compris la logique de base de la machine, il peut retrouver à tout moment la commande nécessaire à ce qu'il veut faire.

- 3. CE QUE VOUS VOYEZ EST CE QUE VOUS AUREZ.
  L'image sur l'écran est une représentation aussi exacte que possible du document qui sera imprimé.

Il existe des feuilles de formatage qui sont en fait des fenêtres contenant des caractéristiques de formatage. L'utilisateur moyen n'a pas besoin d'apprendre le langage des commandes typographiques. Il y a plusieurs sortes de feuilles de formatage :

- pour les caractères (sorte de frappe(police), taille,...)

- pour les paragraphes (indentation, justification, ...)

- pour les tableaux

- pour les entêtes

- pour les documents (longueur de lignes, ...)

L'utilisateur peut appeler à tout moment la feuille de formatage appropriée en sélectionnant l'élément du texte (caractère, paragraphe, tableau, document, etc..) et en poussant la commande appropriée.

Il existe aussi une mise en page automatique qui inclut des pages de deux colonnes.

STAR est capable de montrer tous les symboles d'une équation dans leur position et leur taille exacte telle que l'équation est tapée. De plus, STAR aide automatiquement l'utilisateur. En effet, si l'utilisateur tape le caractère signe de sommation, STAR affichera le sigma sur l'écran et placera le curseur directement sous le signe de sommation pour que l'utilisateur inscrive la limite inférieure de la somme. Cela terminé, en tapant la touche NEXT, STAR placera le curseur au-dessus du signe de sommation afin que l'utilisateur rentre la limite supérieure. En retapant la touche NEXT, STAR replacera le curseur à droite du signe de sommation et l'utilisateur pourra taper l'expression à sommer.

STAR ajuste automatiquement la taille des colonnes du tableau quand l'utilisateur rentre les données dans ces colonnes. Il ne doit donc pas s'occuper de la largeur des différentes

colonnes.

L'écran est assez large pour afficher le contenu complet de deux pages de dimension A4. Si les pages du document sont plus longues que cela, l'utilisateur peut dérouler horizontalement les pages du document pour en voir le contenu entier.

L'écran peut être divisé en 6 fenêtres de toute taille. Les documents peuvent être déroulés horizontalement ou verticalement dans une fenêtre et l'utilisateur peut par exemple prendre une partie d'un document affiché dans une fenêtre et l'insérer dans un document affiché dans une autre fenêtre.

Le langage de programmation du STAR programmation que XEROX appelle Cusp(Customer programming) permet à l'utilisateur d'écrire ses propres programmes d'édition dans un langage très proche de l'anglais. L'utilisateur peut donc créer ses propres routines et inventer un "bouton" sur l'écran qu'il peut sélectionner pour exécuter ses routines.

XEROX ne veut pas vendre STAR comme un système autonome. Toute la puissance de la machine est réalisée dans l'environnement d'un réseau sur lequel un grand nombre de personnes travaillent et ont des accès partagés à des fichiers communs, des imprimantes, etc... STAR est plutôt un produit pour la bureautique qu'un simple système d'éditeur de traitement de texte. Le réseau employé est le réseau "ETHERNET".

6.4. L'éditeur BEEZY de Bobst Graphic. [5]

but de l'éditeur:	texte
affichage:	écran
unité:	caractère
interface:	touche de contrôle/touche de fonction.

Forme des commandes.

Les commandes sont effectuées au moyen de touches de fonction situées sur le clavier. Il faut donner la commande que l'on veut exécuter suivie de la commande "execute". Si on s'aperçoit que l'on s'est trompé de commande, on peut l'annuler en donnant la commande "annule" à condition que la commande que l'on voulait exécuter ne l'aie pas déjà été.

Particularités.

En outre, l'éditeur BEEZY permet :

- de créer un lexique (liste de paragraphes numérotés).
- d'avoir des lettres personnalisées em mélangeant deux fichiers différents, l'un contenant le texte de la lettre et l'autre les adresses des destinataires.

- de sortir directement sur l'imprimante tout ou partie d'un document

- de mettre en page un texte en utilisant les tabulateurs. Ces derniers sont des positions du curseur dans une ligne. Les positions de tabulation sont choisies grâce à la ligne-format qui est une ligne qui ne sert qu'au formatage des lignes de texte par le déplacement des positions de tabulation. On peut toujours modifier une ligne-format en cours de travail.

- de redécouper un texte en pages de longueur en principe égale, à partir de la page affichée et jusqu'à la fin du document. Le nombre de lignes pour une opération de redécoupage est comptée de la même manière que la longueur de la page lors de la sortie sur l'imprimante: avec interprétation des valeurs d'interligne.

6.5. L'éditeur ETUDE. [1]

but de l'éditeur:	document intégré
affichage:	écran
unité:	élément de structure
interface:	touches de contrôle, touche du clavier

Forme des commandes.

Contrairement à STAR et à BRAVO, ETUDE utilise la syntaxe "action / (option) modificateur / objet" où une action peut être la destruction ou le déplacement, un modificateur peut être un nombre ou un mot comme par exemple "next", et un objet peut être un paragraphe, un mot, un document, etc....

Particularités.

Tout en gardant toujours le document formaté mis-à-jour sur l'écran, ETUDE utilise la marge de gauche de l'écran comme une fenêtre de format pour mettre les descripteurs de formatage qui indique le type d'action qui a été utilisée sur une partie particulière du texte.

De cette manière, ETUDE essaye de combler le fossé qui sépare la page non-formatée avec des codes de formatage explicitement exprimés et la page "what you see is what you get" qui, une fois formatée, ne contient pas les informations qui ont provoquées le formatage.

Pour la commande d'aide à l'utilisateur, le système va créer une fenêtre qui montre ce que l'utilisateur était en train de faire et quelles options sont disponibles.

7. Les éditeurs divers.

7.1. Préliminaires

Les éditeurs divers sont des éditeurs qui n'entrent réellement pour une raison ou pour une autre dans aucune des classes précédentes. Ils représentent divers essais pour faire "quelque chose d'autre" que les éditeurs classiques.

7.2. L'éditeur CPT. [1]

but de l'éditeur:	texte
affichage:	écran
unité:	caractère
interface:	clavier/touches de fonction

Forme des commandes.

Les commandes de CPT obéissent à la règle "définir / opérer". Il existe cinq touches pour spécifier un caractère, un mot, une ligne, un paragraphe et une page. On spécifie l'unité sur laquelle vont agir les commandes telles que la destruction, le déplacement, etc...

Particularités.

L'éditeur CPT est un exemple représentatif d'un système de traitement de texte commercial. CPT a été le premier système à offrir un écran blanc pleine page avec des caractères noirs, simulant une feuille de papier sur une machine à écrire. L'affichage est organisé en deux zones : la ligne d'état située au sommet de l'écran et dans laquelle sont affichés le nom du fichier, la position du curseur, les messages d'erreur, etc.... Quelques lignes plus bas, on trouve la ligne de frappe qui reçoit tout ce qu'on frappe.

La touche standard du curseur n'existe pas sur le CPT. A la place, la barre d'espace permet d'avancer dans la ligne de frappe et la touche de "backspace" de reculer. Il n'y a pas besoin de touche pour monter ou pour descendre le curseur car la ligne de frappe ne bouge pas et c'est le document lui-même qui se déplace. Il y a donc naturellement des touches pour monter ou descendre le document. Les marges sont mises en bougeant des marques à gauche et à droite de la ligne de frappe.

CPT procure trois modes d'entrées. Le mode "manuel" simule une machine à écrire; quand l'utilisateur atteind la marge de droite, une sonnette résonne. Le mode "panoramique" procure des retours de chariot automatique quand la marge droite est dépassée. Le mode "indentation" produit une indentation automatique, quand un mot atteind une zone "chaude" définie par le système, en utilisant un algorithme aide par un dictionnaire d'exceptions.

7.3. L'éditeur EMACS. [1] [13]

but de l'éditeur	: texte et programme
affichage	: écran
unité	: caractère
interface	: touches de contrôle/touche du clavier

Forme des commandes.

Généralement, les caractères non imprimables (les séquences de "CTRL" et d'"escape") invoquent les commandes pour modifier le document. Les caractères imprimables sont aussi des commandes; l'action par défaut quand on tape un caractère imprimable est d'insérer ce caractère dans le texte là où se trouve le curseur.

Particularités.

EMACS est un éditeur extensible. En effet, pour étendre ou modifier la fonctionnalité d'EMACS, les utilisateurs écrivent des routines dans le même langage de haut-niveau que celui dans lequel les fonctions standard de l'éditeur sont écrites. Cette possibilité permet aux utilisateurs de franchir les limites imposées par ceux qui ont implémenté l'éditeur. L'utilisateur a la possibilité de mettre beaucoup d'extensions ou de changements dans une librairie qui peut être chargée en mémoire quand on l'invoque. En fait, beaucoup de commandes de base qui existe aujourd'hui étaient à l'origine des extensions écrites par des utilisateurs et qui furent adoptées plus tard dans le système.

Les constructeurs d'EMACS ont divisé le système en deux langages : le langage d'édition (EMACS) et le langage de programmation (TECO). Le langage d'édition est écrit dans le langage de programmation. Le langage d'édition agit comme un interpréteur de commandes et fait tourner le langage de programmation. Les descendants d'EMACS utilisent LISP comme langage de programmation, ajoutant plus d'efficacité et de lisibilité aux programmes TECO, caractérisés par leur obscurité.

8. Le système Smalltalk-80. [8]

8.1. Préliminaires

En regardant les secrétaires apprendre à utiliser les éditeurs de textes, on peut se convaincre que les ordinateurs ne sont pas aussi agréables qu'on pourrait le croire. La question la plus courante posée par les nouveaux utilisateurs, qui avant était "Comment puis-je faire cela ?" est maintenant "Comment sortir de ce mode ?".

Les nouveaux venus ne sont pas les seules victimes des modes. Les experts tapent souvent des commandes utilisées dans un mode quand ils sont dans un autre; cela menant à des conséquences indésirables et parfois même désastreuses. Dans beaucoup de systèmes, taper la lettre "D" peut avoir des significations aussi diverses que "remplacer le caractère sélectionné par D", "insérer un D avant le caractère sélectionné" ou "détruire le caractère sélectionné".

Quand on utilise un éditeur de texte, on ne peut pas obtenir une liste des fichiers sur le disque. De même, si on stoppe un programme et qu'on en commence un autre, les données affichées par le premier programme sont probablement éliminées de l'écran et irrémédiablement hors de vue. En général, exécuter un programme dans la plupart des systèmes nous met dans un mode où les commandes des autre programmes nous sont inaccessibles. Dan Swinehart appelle cela le dilemme de la préemption. En effet, le système nous force à choisir entre les différents modes et on ne peut pas en avoir plusieurs à la fois.

Alan Kay a imaginé un principe pour éliminer les différentes sortes de modes, ces sortes causant ce qu'il appelle "le dilemme de la préemtion". Ce principe se nomme "le chevauchement de fenêtres".

Une notion de plus en plus courante est celle de fenêtre. Dans certains systèmes, on peut avoir plusieurs tâches en route,chacune représentée dans une fenêtre différente et on peut voyager librement entre les tâches en voyageant entre les fenêtres. L'idée de Kay était de permettre un accès sélectif à plusieurs fenêtres même partiellement cachées. L'écran est représenté comme la surface d'un bureau et les fenêtres comme des feuilles de papier se chevauchant. A l'aide d'un moyen de pointage (flèche, souris, palpeur....) qui bouge un curseur sur l'écran, on peut bouger le curseur sur une feuille partiellement couverte et presser un bouton sur le moyen de pointage pour découvrir la feuille.

Les avantages du principe du chevauchement des fenêtres sont :

- Les affichages associés à plusieurs tâches de l'utilisateur peuvent être vus simultanément.
- Voyager parmi les tâches est fait en pressant un bouton.
- Aucune information n'est perdue en voyageant parmi les tâches.
- L'espace de l'écran est utilisé économiquement.

8.2. Smalltalk 80

but de l'éditeur	: document
affichage	: écran
unité	: élément de structure
interface	: touche de fonction / souris

Forme des commandes.

La forme des commandes n'était pas spécifiée dans la documentation qui était à notre disposition.

Particularités.

Chaque fenêtre a un menu qui comprend des commandes pour réencadrer la fenêtre autre part avec une nouvelle taille, pour fermer la fenêtre, pour imprimer le contenu d'une fenêtre sur une imprimante et pour retrouver les fenêtres cachées par d'autres.

Il existe une commande qui renverse l'effet de la dernière commande exécutée. Bien que le but principal d'"undo" est de compenser le manque de commande de confirmation, il peut aussi être utilisé pour changer d'avis après avoir lancé une commande.

Pour réveiller une autre fenêtre, on bouge le curseur sur une nouvelle fenêtre et on presse le bouton de sélection sur la souris. Quand une fenêtre est réveillée, elle est entièrement affichée et elle n'est plus couverte par les autres fenêtres.

Bien que les fenêtres se chevauchant nous rendent capables de garder l'état de plusieurs tâches sur l'écran en même temps, on peut parfois vouloir travailler sur plusieurs projets entièrement différents, chacun ayant plusieurs tâches. Smalltalk nous laisse avoir un "dessus de bureau" différent pour chaque projet. Sur chaque "dessus de bureau", on a les fenêtres pour les tâches concernant ce projet. Pour nous aider à nous déplacer d'un bureau à l'autre, un "dessus de bureau" peut avoir plusieurs fenêtres de projet qui nous montre les autres "dessus de bureau" disponibles et nous permet de passer de l'un à l'autre.

Une commande importante est le "snapshot". L'état entier de l'environnement du Smalltalk _ comprenant les données, les traitements en cours et qui sont suspendus, les fenêtres sur l'écran et les projets de "dessus de bureau" _ peut être momentanément gelé et sauvé en mémoire secondaire. Le "snapshot" peut être restauré plus tard pour se retrouver exactement dans les conditions ou on se trouvait au moment du sauvetage en catastrophe.

D'un point de vue de mémoire centrale, on doit disposer d'au moins 256 Kbytes car tout l'environnement intégré se trouve sur un espace adresse. Il inclut en effet non seulement le support de langage d'exécution mais aussi les fenêtres orientées graphiques, l'éditeur, le compilateur et d'autres commandes du logiciel. Le package logiciel procure :

- la sortie pour l'utilisateur à travers des fenêtres se chevauchant.

- l'entrée à partir d'un clavier, d'une souris et des menus.

- le traitement uniforme de textes, graphiques, symboles et nombres.

9. Conclusions

Nous regroupons les principales caractéristiques de ces éditeurs dans deux tableaux. Ces derniers ne comportent évidemment pas toutes les caractéristiques de tous les éditeurs mais nous pensons qu'ils donnent un bon résumé de cette classification.

CARACT. / EDITEUR	but	affichage	unité	interface	souris	écran A4
CMS	prog. texte	ligne	ligne	mots à taper	non	non
ED	prog. texte	ligne	ligne	mots à taper	non	non
BB	texte prog.	écran	caract.	touche fct touche ctrl	non	non
Z	texte prog.	écran	caract.	touche fct touche ctrl	non	non
EDIS	prog.	écran	symbole	clavier	non	non
PASCAL	texte prog.	écran	caract.	touche du clavier	non	non
THOT	texte	écran	élément struct.	touche fct	non	oui
PEN	docum.	écran	élément struct.	non-spécifié	non	?
BRAVO	texte	écran	élément struct.	touche ctrl souris	oui	?
STAR	texte	écran	élément struct.	touche ctrl souris	oui	double
BEEZY	texte	écran	élément struct.	touche ctrl touche fct	non	oui
ETUDE	docum.	écran	élément struct.	touche ctrl touche curs.	non	oui
CPT	texte	écran	caract.	touche fct clavier	non	non
EMACS	texte prog.	écran	caract.	touche ctrl	non	non
SMALLTALK	docum.	écran	élément struct.	touche fct souris	oui	oui
EDIF	Texte prog	écran	caract	Touche claire	non (oui)	non

CARACT. / EDITEUR	undo *	justification	fenêtre	analyse **	snapshot ***	indent
CMS	non	non	non	non	non	non
ED	non	non	non	non	non	non
EDIS	non	non	non	oui	non	oui
BB	non	non	oui	non	oui	non
Z	non	oui	non	non	non	non
PEN (BBN)	non	non	oui	non	non	non
PASCAL	non	non	non	non	non	oui
BRAVO	oui	oui	oui	non	non	non
STAR	oui	oui	oui	non	non	non
BEEZY	non	oui	non	non	non	non
ETUDE	non	oui	non	non	non	non
THOT	non	oui	non	non	non	non
WALKER	non	non	non	non	non	non
PEN(YALE)	non	non	non	non	non	non
CPT	non	non	non	non	non	oui
EMACS	non	non	non	non	non	non
SMALLTALK	oui	oui	oui	non	oui	non
edif	oui	oui	oui	non	oui ?	non.

* undo : commande qui annule l'effet de la commande précédente

** analyse : commande qui permet l'analyse syntaxique d'un programme

*** snapshot : fct qui, en cas de panne de l'ordinateur, sauve tous les fichiers et permet de retrouver le travail interrompu là où il était avant la panne.

L'impression majeure que l'on retire de l'étude que nous venons de présenter est que l'évolution des besoins en traitement de textes, l'augmentation de l'expérience accumulée dans la conception des éditeurs et le développement de technologies nouvelles se conjuguent et convergent vers l'apparition de systèmes de traitement de documents puissants, agréables à utiliser et versatiles.

Les caractéristiques essentielles des systèmes actuellement en développement sont :

- les capacités de formattage en cours d'édition
- la concordance entre l'image fournie par l'écran et le résultat imprimé
- la vision simultanée de plusieurs aspects d'un document ou de plusieurs documents par le biais des "fenêtres".

Chapitre 2

ANALYSE FONCTIONNELLE

Chapitre 2: Analyse fonctionnelle

(Françoise Fourny)

1. Introduction

Dans le cadre de notre mémoire, nous avons élaboré un système de traitement de documents dont les caractéristiques générales ont été définies lors du stage que nous avons effectué dans le laboratoire de micro-informatique de l'Ecole Polytechnique Fédérale de Lausanne.

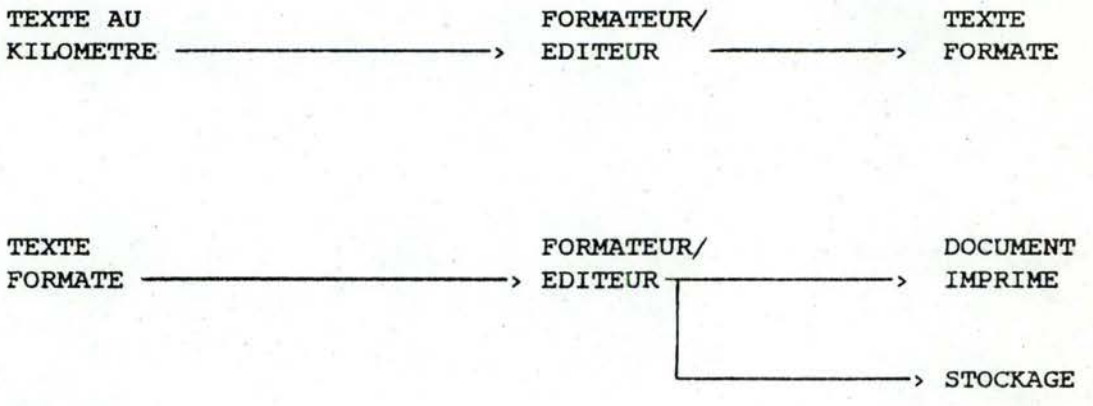
Le système proposé est un système de traitement de textes qui, dans la suite, peut s'adapter assez facilement pour devenir un système de traitement de documents.

Ce système appartient à la classe des sytèmes dits "graphiques" et s'inspire de STAR de XEROX. Pourquoi avoir choisi cette orientation ? Il nous a semblé que le STAR possédait les caractéristiques nécessaires à un bon système de traitement de textes c'est-à-dire une facilité d'utilisation pour un utilisateur non-initié et la possibilité de présenter à l'écran un document dans une forme assez proche de ce qui sera imprimé(What you see is what you get). Le système que nous avons réalisé se différencie du STAR par deux aspects. Il ne comporte que la partie traitement de textes alors que le STAR est un système de traitement de documents complet. De plus, la gestion des fenêtres dans le système que nous avons implémenté est moins complète que dans le STAR.

Pour réaliser cela, nous avons défini les propriétés suivantes:

- Le système travaillera sur un texte formatté ou au kilomètre (c-à-d texte entré sans retour de chariot pour la fin de ligne). Le système devra donc comporter des primitives de formatage (voir schéma 1).

Schéma 1.



- Le système est un éditeur à fenêtres où la majorité des commandes disponibles apparait sous la forme de menus. Les menus du mode où se trouve l'utilisateur apparaissent en permanence sur l'écran. Une fenêtre est un emplacement de l'écran sur lequel est visualiée une zone déplaçable du document. La fenêtre ne visualise pas nécessairement la zone déplaçable sur toute la largeur. On remarque donc que les mouvements dans le document se font de haut en bas et que pour avoir les mouvements de gauche à droite il suffit de déplacer le zone déplaçable sous la fenêtre.

- Le système a accès à plusieurs documents en une session et en même temps. Cela implique donc que nous pouvons avoir plusieurs fenêtres. Plusieurs fenêtres peuvent visualiser soit plusieurs documents différents, soit plusieurs zones différentes d'un même document, soit encore plusieurs versions d'un même document.

- Le système fait partie de la classe "WHAT YOU SEE IS WHAT YOU GET".

- Le système doit comporter un moyen rapide de sortie qui permet de pouvoir se retrouver, plus tard, dans la situation interrompue.

- Le système, pour être facile d'emploi, comportera un dispositif souris (voir annexe) pour commander l'éditeur mais gardera la possibilité d'utiliser le clavier. Cet éditeur sera implémenté avec deux variantes:

  - soit le déplacement du curseur par la souris.

  - soit le déplacement du curseur par le clavier.

  Dans la variante avec souris, l'utilisateur déplace le curseur au moyen de celle-ci et sélectionne l'opération désirée grâce au bouton de sélection.

  Dans la variante avec clavier, l'utilisateur se déplace grâce à des touches du clavier et sélectionne par une touche.

  Les avantages résultant de ces choix sont la clarté et la facilité d'utilisation de cet outil de traitement de textes. L'utilisateur sait à tout moment où il en est dans l'arbre de l'éditeur grâce à la disposition de l'écran qui rappelle continuellement la branche de l'éditeur sur laquelle il se trouve et lui permet de connaître l'ensemble des opérations possibles dans le mode choisi.

Expliquons maintenant le démarche d'analyse qui suivit le définition de ces caractéristiques. L'analyse complète d'un projet informatique comporte trois phases :

- l'analyse conceptuelle ou l'analyse d'opportunité.
- l'analyse fonctionnelle.
- l'analyse organique.

Un mot sur l'analyse d'opportunité de problèmes du type éditeur et système de traitement de textes. De nombreuses enquêtes auprès des secrétaires ont été faites par des constructeurs afin d'étudier les besoins en matière de traitement de textes. Toutes les fonctions possibles d'un éditeur ont même fait l'objet d'une synthèse SOBEMAP[17]. Une étude d'efficacité réalisée par SIEMENS montre qu'il y a un gain d'efficacité d'environ 50 pourcents en utilisant des systèmes de traitement de textes. De multiples enquêtes faites chez des utilisateurs indiquent des gains de l'ordre de 20 à 300 pourcents.

L'analyse fonctionnelle commence quand l'analyse d'opportunité a été réalisée. La démarche utilisée pour l'analyse fonctionnelle consiste à définir QUE FAIRE sans s'occuper du COMMENT FAIRE.

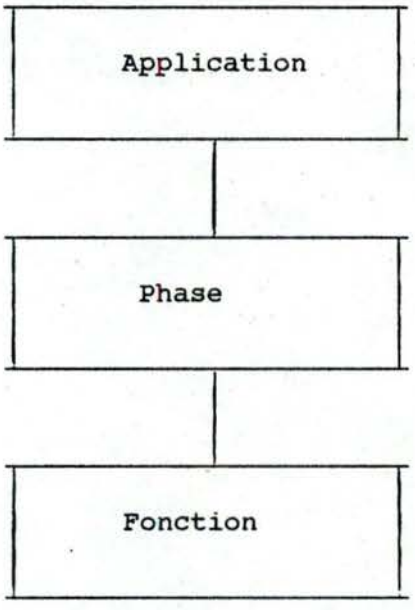
L'analyse fonctionnelle se présente sous deux angles différents : la structuration et la spécification des traitements à réaliser et la spécification des objets ou données sur lesquels ces traitements agissent.

2. Spécifications fonctionnelles.

2.1. Spécification de la structure des traitements.

2.1.1. Rappel de la hiérarchie utilisée.

Le principe utilisé lors de la description des traitements est celui de décomposition hiérarchique suivante:



Expliquons quelque peu les différents concepts de cette hiérarchie. L'"application" regroupe un ensemble de phases relatives à un flux homogène d'informations ayant une permanence dans l'organisation.

La "phase" est un traitement manuel ou automatisable possédant une unité spatio-temporelle. Cette unité d'exécution implique que la phase soit entièrement exécutée dans une cellule d'activité. Une cellule d'activité est un centre d'activité homogène dans le temps et l'espace, doté de ressources humaines et/ou matérielles et pourvue de règles de comportement nécessaires à son fonctionnement. Une phase se compose de fonctions.

Une "fonction" représente des traitements considérés comme élémentaires. Elle est associée à un objectif et à un comportement considérés comme élémentaires dans l'organisation.

2.1.2. Structure générale de l'éditeur.

Dans cette partie du mémoire, nous présentons la structure générale de l'éditeur que nous avons réalisé.

Le suffixe -appl désigne une application.
Le suffixe -ph désigne une phase.
Le suffixe -fct désigne une fonction.

On peut donc voir dans cette présentation la découpe hiérarchique dont le rappel a été fait plus haut.

```
Editeur-appl
        Création_modification-ph
                lecture-fct
                sauvetage-fct
                format-fct
                insertion-fct
                backspace-fct
                delete-fct
                justification-fct
                pagine-fct
                saut-de-page-fct
                visualisation-des-caractères-spéciaux-fct
                défaire-fct
                encadre-fct
                efface-fct
                transfert-fct
                montre-fct
                copie-fct
                centre-fct
                recherche-fct
                change-fct
                caractère-fct
                minuscule-fct
                majuscule-fct
                déplacement-un-bas-fct
                déplacement-un-haut-fct
                déplacement-quatre-bas-fct
                déplacement-quatre-haut-fct
                déplacement-début-de texte-fct
                déplacement-fin-de texte-fct
                déplacement-car-droit-fct
                déplacement-car-gauche-fct
                déplacement-mot-droit-fct
                déplacement-mot-gauche-fct
                retour-fct
        Destruction-ph
                destruction-phys-fct
                destruction-log-fct
                activation-fct
        Impression-ph
                impression-fct
                stop-fct
```

2.1.3. Spécification des traitements

Application : Editeur

Le système de préparation de documents est un système qui synthétise les éditeurs graphiques, de textes, formateurs et qui en combine et intègre les utilitaires. L'édition de documents est le moyen interactif pour créer, réviser, corriger et conserver des documents.

Ce système a été conçu pour éditer des documents qui, après transformation par l'éditeur, devront être imprimés sur du papier de dimension A4. Cet éditeur ne peut éditer des documents qui n'ont pas été créés par lui-même. Cette restriction est assez classique.

Phase : Création_modification-ph-2

La phase d'édition permet d'effectuer toutes les opérations d'édition de textes possibles. C'est-à-dire que, dans cette phase, on va retrouver les différentes actions possibles sur un texte. On va pouvoir créer un nouveau document (création), modifier un document qui existait déjà (modification). L'utilisateur doit introduire un nom de document correct. Le cas de la création d'un document est absolument identique au cas de la modification d'un document seule change la condition d'entrée : dans le cas de la création le nom de document que l'utilisateur entre ne peut déjà exister, tandis que dans le cas de la modification, il doit exister.

Dans le cas d'une création, il s'agit de créer un nouveau document. Si le nom du document entré par l'utilisateur est valide, le texte proprement dit peut être créé. Pour cela, l'utilisateur doit disposer de primitives d'actions sur le texte. Ces primitives ou fonctions permettent d'une certaine manière de définir une partie des caractéristiques du texte. Ces primitives d'action sur le texte sont :

- insérer
- Recul arrière
- Suppression
- déplacement d'une ligne vers la fin du texte
- déplacement d'une ligne vers le début du texte
- déplacement de quatre lignes vers le début du texte
- déplacement de quatre lignes vers la fin du texte
- déplacement au début du texte
- déplacement à la fin du texte
- déplacement d'un caractère vers la gauche dans le texte
- déplacement d'un caractère vers la droite dans le texte
- déplacement d'un mot vers la gauche dans une ligne du texte
- déplacement d'un mot vers la droite dans une ligne du texte
- justification
- pagination
- saut de page
- visualisation des caractères spéciaux
- montre
- copie
- cherche
- change
- minuscule
- majuscule
- transfert
- caractère
- défaire
- encadre
- efface
- centre
- retour

Dans le cas d'une modification de texte, les

fonctions disponibles sont exactement les mêmes que dans le cas d'une création; seule varie la condition d'entrée.

D'autres fonctions sont encore disponibles :
- en cas de modification, il faut pouvoir lire le texte (fonction lecture).
- en cas de modification ou création, il faut formatter le texte (fonction format).
- en cas de modification ou de création, tout ce qui a été crée ou modifié doit être sauvé (fonction sauvetage).

Fonction : lecture-fct-2

la fonction de lecture permet deux actions :
- visualiser le texte dont le nom valide a été entré.
- se déplacer dans le texte.

Les déplacements permis sont :
- déplacement d'une ligne vers la fin du texte
- déplacement d'une ligne vers le début du texte
- déplacement de quatre lignes vers le début du texte
- déplacement de quatre lignes vers la fin du texte
- déplacement au début du texte
- déplacement à la fin du texte
- déplacement d'un caractère vers la gauche dans le texte
- déplacement d'un caractère vers la droite dans le texte
- déplacement d'un mot vers la gauche dans une ligne du texte
- déplacement d'un mot vers la droite dans une ligne du texte

Ces fonctions de déplacements sont définies ultérieurement.

Fonction : sauvetage-fct

La fonction de sauvetage est la fonction qui permet de mémoriser les modifications subies par un texte soit lors de sa création, soit lors de sa modification, soit encore lors de son formatage. La fonction de sauvetage physique effectue la mémorisation du texte dans le répertoire de l'utilisateur.

Fonction : format-fct-2

La fonction format va permettre de définir un deuxième type de caractéristiques du texte. Ce sont les caractéristiques d'impression du texte. Ces caractéristiques sont :
- largeur de la marge à gauche
- longueur de l'espace précédant le texte en début de page
- longueur du texte sur la page
- interligne
- police de caractères utilisée pour imprimer le texte
- largeur du texte sur la feuille

- positionnement des tabulateurs
- justification à gauche
- justification à gauche et à droite
- césure des mots
- vérification de la césure
- nombre maximun de blancs entre 2 mots

Fonction : insertion-fct

L'insertion permet d'insérer un seul caractère dans le texte. Il suffit de se positionner à l'endroit où il faut insérer et commencer à introduire le(s) caractère(s) à ajouter au texte.

Fonction : recul-arrière-fct

La fonction de recul arrière permet de reculer dans le texte d'un caractère vers la gauche quand on est au milieu d'une ligne ou bien à la fin de la ligne précédente lorsqu'on est au début d'une ligne. Cette fonction est traditionnellement réalisée par une touche du clavier.

Fonction : delete-fct

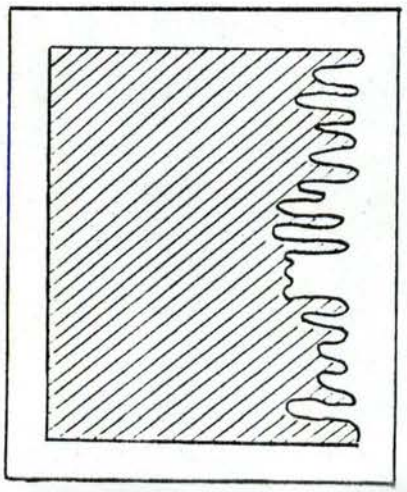
Le delete est une fonction permettant d'effacer le caractère courant. La fonction delete est, souvent, comme le backspace, réalisée par une touche du clavier.

Fonction : justification-fct

La fonction de justification est une fonction qui définit une présentation du texte. Il existe deux types de justifications :

- à gauche.
- à gauche et à droite.

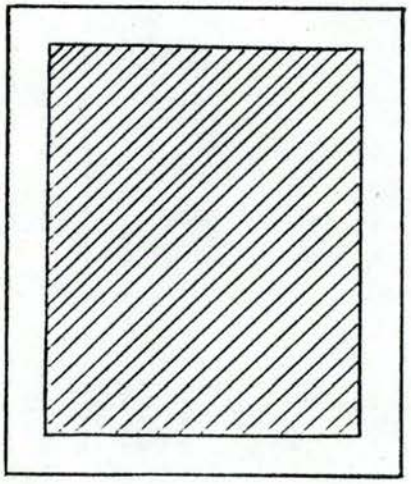
La justification à gauche consiste en un alignement rectiligne du texte sur la marge gauche définie dans la fonction format. Chaque ligne ne dépasse jamais la largeur maximale définie dans la fonction format-fct-2. Quand l'ajoute d'un mot peut causer le dépassement de cette limite, le mot est rejeté à la ligne suivante. Une ligne du texte est donc terminée soit quand on atteint la largeur maximale soit quand l'utilisateur a déterminé explicitement qu'il désire aller à la ligne par un caractère de fin de ligne(CR) ou un caractère de saut de page. Le résultat de cette justification est un texte en drapeau pareil au schéma qui suit.



texte en drapeau

La justification à gauche et à droite consiste en un alignement rectiligne du texte sur la marge gauche pour le début de chaque ligne et un alignement rectiligne à droite sur la largeur de texte. Quand, dans la constitution d'une ligne, l'ajoute d'un mot cause le dépassement de la largeur limite du texte, le mot est non seulement rejeté à la ligne suivante mais la longueur restante (la différence entre la longueur réelle de la ligne et la longueur maximale) est répartie entre chaque mot de la ligne afin d'avoir une ligne s'ajustant parfaitement à la longueur de la ligne maximale définie par l'utilisateur.

Si le nombre de blancs entre deux mots est trop grand, une autre optique est d'effectuer la césure du mot qui dépasse. Cette césure est effectuée selon les règles prédéfinies par la grammaire de la langue française. Ensuite, après la césure, il faut encore répartir l'espace restant, s'il y en a, entre chaque mot de la ligne.



Fonction : pagine-fct

La fonction de pagination est une fonction permettant d'ajouter automatiquement un numéro de page à chaque page du texte édité par l'utilisateur. La visualisation des numéros de page ne se fait qu'à l'impression.

Fonction : saut-de-page-fct

La fonction saut de page permet à l'utilisateur de spécifier explicitement qu'il désire passer à la page.

Fonction : visualisation-des-caractères-spéciaux-fct

La fonction de visualisation des caractères spéciaux permet de voir les caractères ajoutés par le programme éditeur lors des différentes actions effectuées sur le texte.

Fonction : défaire-fct

La fonction défaire permet de retrouver le texte tel

qu'il était avant l'action de la dernière primitive. Cette fonction est utile car elle permet un retour en arrière lorsque l'action effectuée sur le texte ne satisfait pas l'utilisateur.

Fonction : encadre-fct

La fonction encadre permet de tracer un encadrement autour d'une portion de texte délimitée par l'utilisateur.

Fonction : efface-fct

La fonction efface permet d'effacer une portion de texte délimitée par l'utilisateur. La longueur de cette portion de texte est illimitée.

Fonction : transfert-fct

La fonction de transfert permet à l'utilisateur de transférer une portion de texte délimité en un autre endroit défini par l'utilisateur. La portion de texte transférée a une grandeur illimitée et l'endroit de transfert peut se trouver n'importe où dans le texte (sur la même page ou sur une ou plusieurs autres).

Fonction : montre-fct

La fonction montre effectue la visualisation d'une portion de texte que l'utilisateur a mémorisée. La longueur de la portion de texte est illimitée.

Fonction : copie-fct

La fonction copie effectue la mémorisation d'une portion de texte définie par l'utilisateur ou bien effectue le transfert d'une portion de texte mémorisée dans un autre texte à l'endroit défini par l'utilisateur.

Fonction : Centre-fct

La fonction centre effectue le centrage d'une ligne de texte par rapport à la largeur maximale du texte. Cette fonction de centrage est utile pour centrer les titres.

Fonction : recherche-fct

La fonction recherche permet de rechercher dans un texte toutes les occurrences d'une chaîne de caractères définie par l'utilisateur.

Fonction : change-fct

La fonction change effectue comme la fonction de recherche la recherche de toutes les occurrences d'une chaîne de caractères dans un texte mais en plus effectue, si l'utilisateur le désire, la substitution de cette chaîne par une autre que l'utilisateur a définie.

Fonction : caractère-fct

La fonction caractère permet de changer la police des caractères pour une partie du texte afin de mettre du relief dans celui-ci. Les possibilites de choix sont les suivantes :

- Helvesan
- Messenger
- Grec
- Italique
- Souligné
- Surligné
- Gras

Fonction : minuscule-fct

La fonction minuscule permet de substituer tous les caractères majuscules en des caractères minuscules dans la portion de texte définie par l'utilisateur.

Fonction : majuscule-fct

La fonction majuscule permet de substituer tous les caractères minuscules en des caractères majuscules dans la portion de texte définie par l'utilisateur.

Fonction : déplacement-un-bas-fct

La fonction de déplacement-un-bas permet à l'utilisateur d'avancer dans le texte d'une ligne à la fois.

Fonction : déplacement-un-haut-fct

La fonction de déplacement-un-haut permet à l'utilisateur de reculer dans le texte d'une ligne à la fois.

Fonction : déplacement-quatre-bas-fct

La fonction de déplacement-quatre-bas permet à l'utilisateur d'avancer dans le texte de quatre lignes à la fois. Le déplacement de quatre lignes à la fois dans un texte est utile si l'on désire parcourir rapidement le texte. Le nombre 4 est arbitrairement choisi afin d'éviter un défilement trop rapide du texte.

Fonction : déplacement-quatre-haut-fct

La fonction de déplacement-quatre-haut permet à l'utilisateur de reculer dans le texte de quatre lignes à la fois.

Fonction : déplacement-début-de-texte-fct

La fonction de déplacement-début-de-texte permet à l'utilisateur de se déplacer au début du texte.

Fonction : déplacement-fin-de-texte-fct

La fonction de déplacement-fin-de-texte permet à l'utilisateur de se déplacer en fin du texte.

Fonction : déplacement-car-gauche-fct

La fonction de déplacement-car-gauche permet à l'utilisateur de se déplacer dans une ligne d'un caractère vers la gauche. Si l'utilisateur se trouve en début de ligne, on passe à la fin de la ligne précédante.

Fonction : déplacement-car-droit-fct

La fonction de déplacement-car-droit permet à l'utilisateur de se déplacer dans une ligne d'un caractère vers la droite. Si l'utilisateur se trouve en fin de ligne, on passe au début de la ligne suivante.

Fonction : déplacement-mot-gauche-fct

La fonction déplacement-mot-gauche permet à l'utilisateur de se déplacer de un mot vers la gauche dans une ligne. Lorsque l'utilisateur est en début de ligne, le déplacement continue à la ligne précédante.

Fonction : déplacement-mot-droit-fct

La fonction déplacement-mot-droit permet à l'utilisateur de se déplacer de un mot vers la droite dans une ligne. Lorsque l'utilisateur est en fin de ligne, le déplacement continue à la ligne suivante.

fonction : retour-fct

La fonction retour permet à l'utilisateur de quitter la phase création_modification du système.

Phase : Destruction-ph-3.

La phase de destruction est comme son nom l'indique une phase où toutes les opérations relatives à la destruction d'un texte s'exécutent. On y retrouve les différentes fonctions suivantes :

- la destruction logique d'un texte.
- la réactivation d'un texte.
- la lecture d'un texte afin de vérifier si c'est le bon texte que l'on veut détruire. Il s'agit de la même fonction que dans la phase d'édition.
- la destruction physique d'un texte.

Fonction : Destruction-phys-fct

La fonction de destruction physique d'un texte est l'effacement total d'un texte du répertoire de l'utilisateur.

Fonction : Destruction-log-fct

La fonction de destruction est la fonction qui va permettre de détruire logiquement un texte. C'est-à-dire que cette fonction va mémoriser que l'utilisateur a décidé d'effacer un texte particulier mais ne le détruit pas réellement à ce moment. Malgré cette destruction logique, l'utilisateur peut encore accéder à ce texte car le texte ne sera détruit physiquement qu'en sortant de l'éditeur.

Pour détruire logiquement, l'utilisateur va donner un nom de texte valide. Le texte à détruire doit avoir été créé ou modifié au cours de la session.

Fonction : reactivation-fct

La fonction de réactivation est la fonction qui a comme objectif de permettre de récupérer un texte qui a été détruit logiquement dans la session de l'éditeur. Cette fonction est l'inverse de la fonction de destruction logique. L'utilisateur peut donc travailler à nouveau sur ce texte comme s'il n'avait pas été détruit. L'utilisateur doit donner un nom de texte valide et qui a été crée ou modifié au cours de la session.

Phase : Impression-ph-1

La phase d'impression permet d'imprimer un ou plusieurs documents selon les caractèristiques prédéfinies lors de la phase d'édition (édition-ph-2).

Cette phase comporte deux fonctions : l'impression du document proprement dite et une fonction d'arrêt de la phase d'impression.

Fonction : Impression-fct-1

La fonction d'impression est la fonction qui va permettre l'impression du document dont le nom est connu et valide sur papier A4. Plusieurs possibilités doivent être offerte :

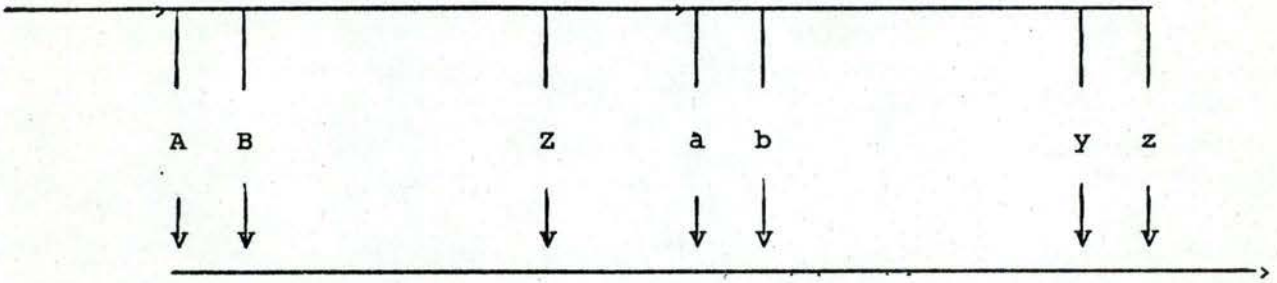
- soit l'impression du document en entier
- soit l'impression d'une page du document

Fonction : stop-fct-1

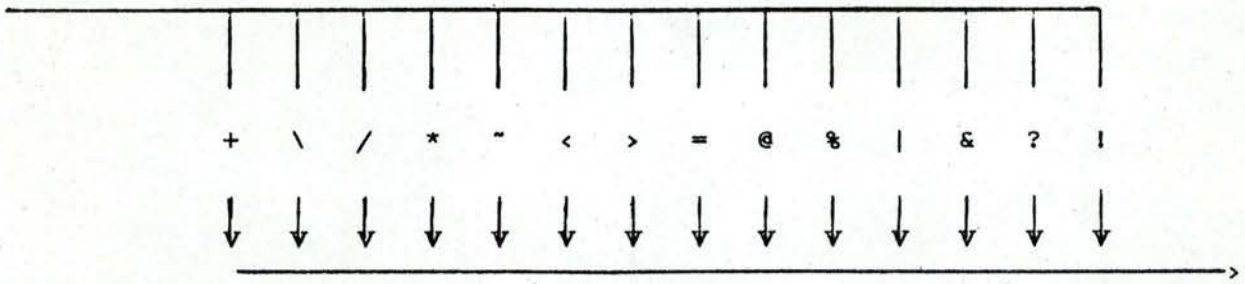
La fonction STOP permet la sortie de la phase d'impression. On ne peut, toutefois, arrêter une impression en cours par cette fonction.

2.2. Dictionnaire des données.

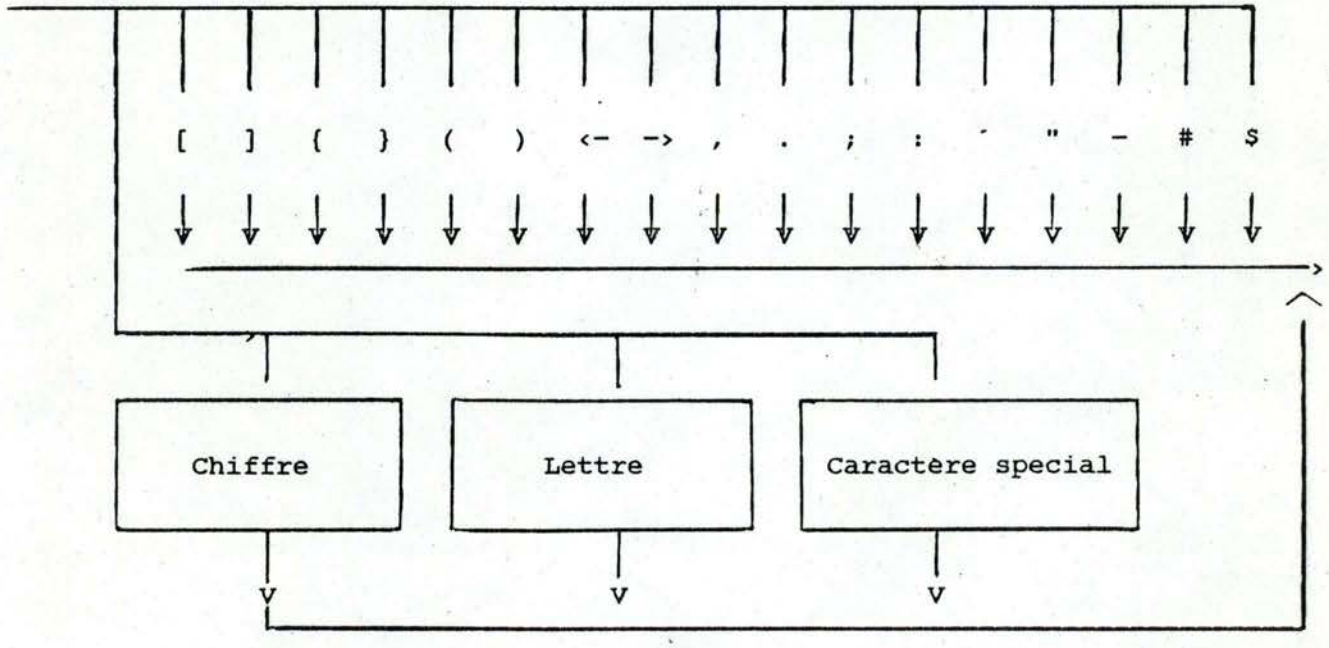
Lettre:



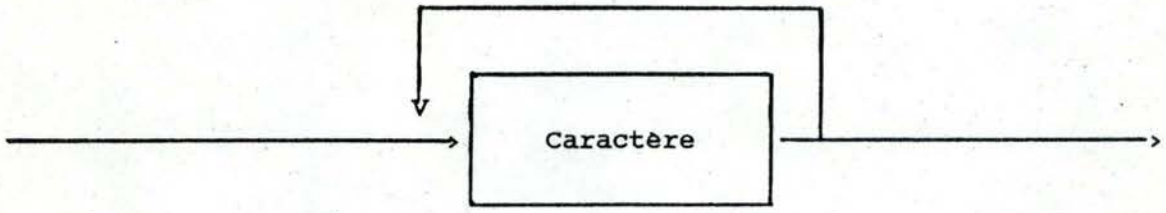
Caractère spécial :



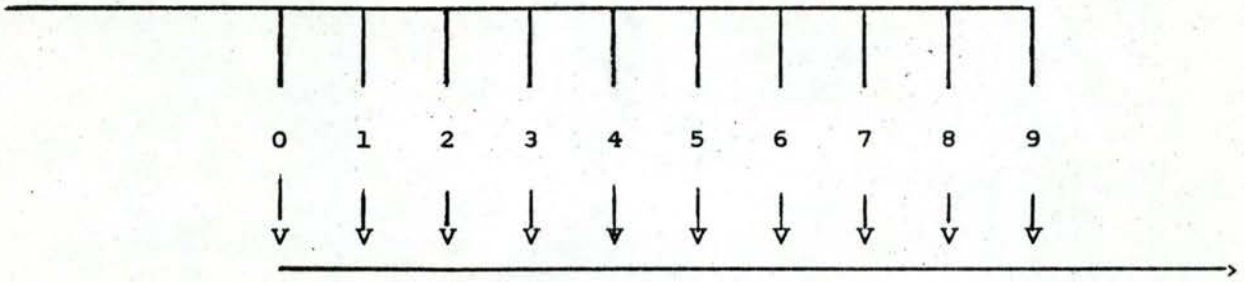
Caractère :



Chaîne de caractères :

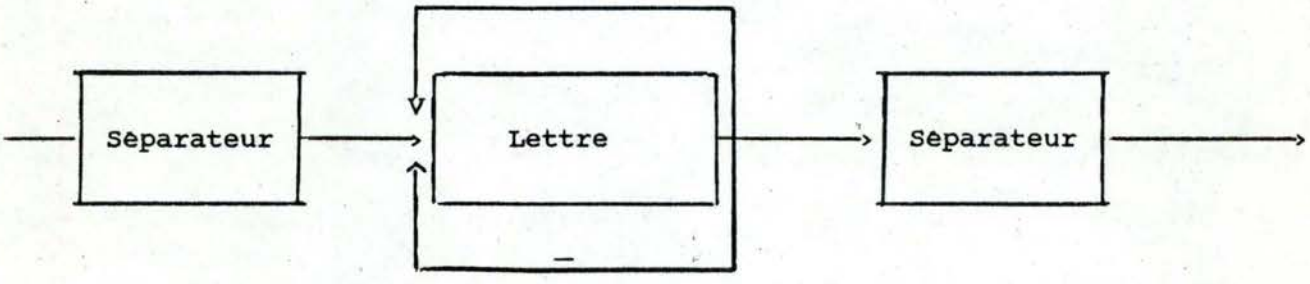


Chiffre :

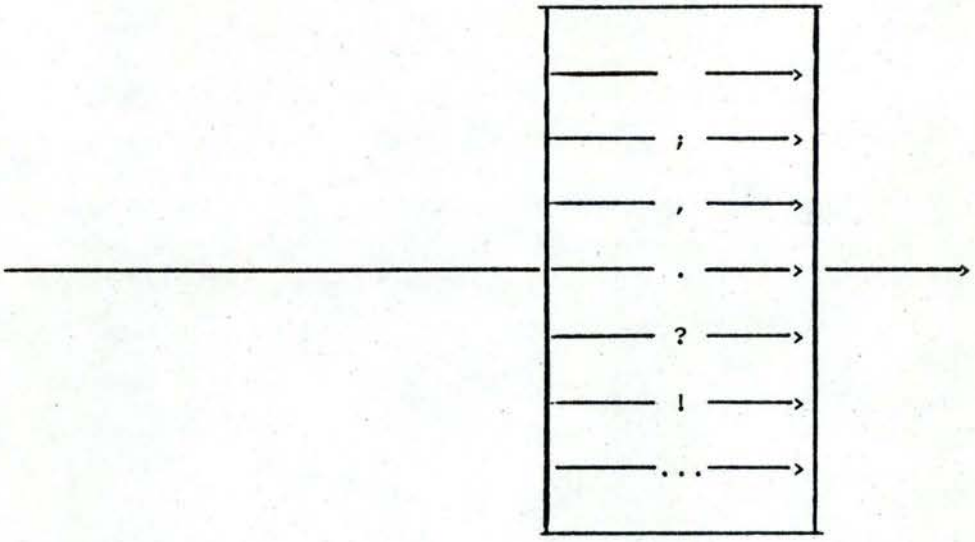


Mot :

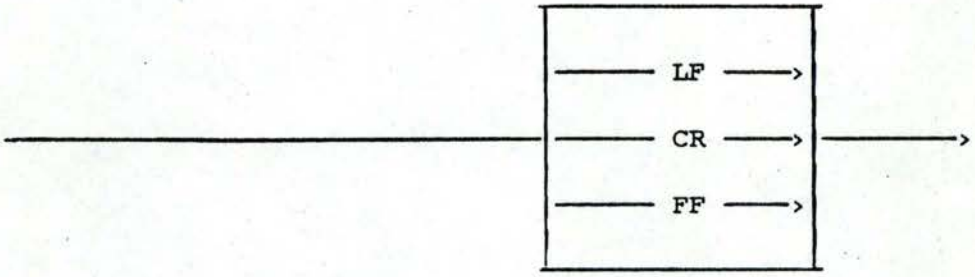
La caractéristique principale d'un mot est : Césure: coupure du mot. Les deux solutions possibles sont : couper le mot ou ne pas couper le mot.



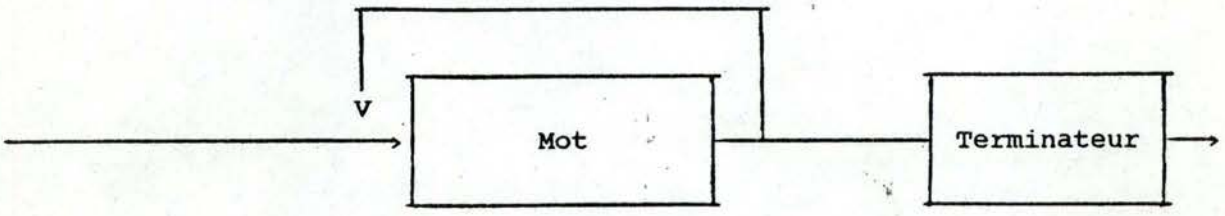
Separateur :



Terminateur :

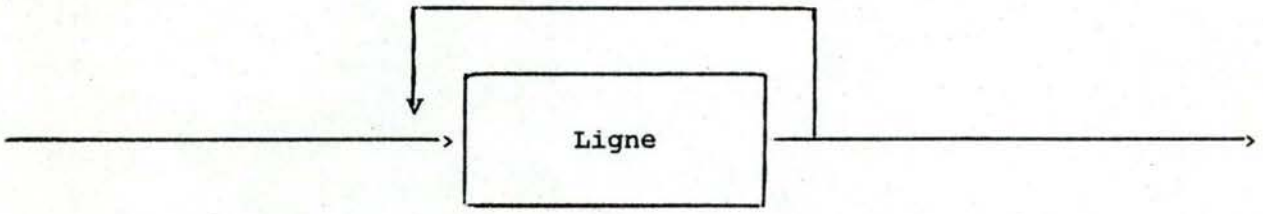


Ligne :



Page :

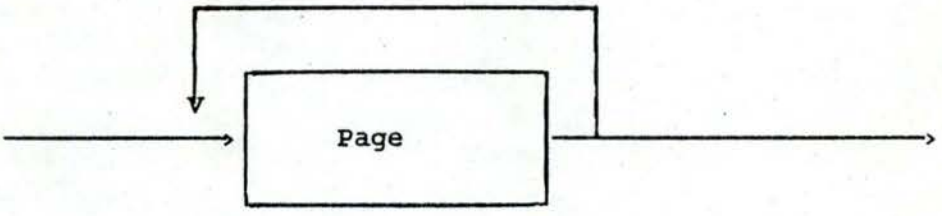
Une page est caractèrisée par un numèro.



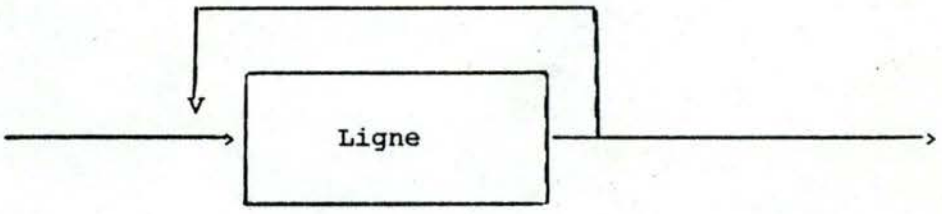
Texte :

Un texte peut avoir trois définitions différentes

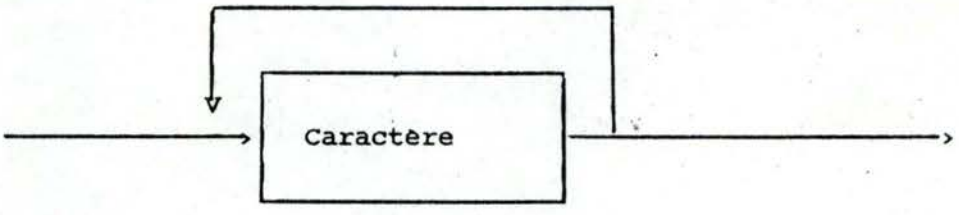
1)



2)



3)



Les caractèristiques d'un texte sont les suivantes :

- Nom : chaîne de caractères de 14 caractères de long .
- Date de crèation du texte : date sous la forme (JJ MMM AA HH:mm).

- Date de mise-à-jour : date sous le forme (JJ MMM AA HH:mm).

- Marge gauche : nombre de caractères blancs à gauche d'un texte. La marge gauche doit être comprise entre 0 et 100 caractères blancs.

- Largeur de texte : nombre de caractères que comportera une ligne du texte. La largeur du texte doit être comprise entre 0 et 100 caractères.

- Interligne : espace qui est entre deux lignes écrites ou imprimées. L'interligne doit être compris entre 1 et 4. Il y a donc quatre interlignes possibles comme sur la plupart des machines à écrire.

- Marge haut : nombre de lignes blanches en haut d'un texte. La marge haut doit être comprise entre 0 et 66 lignes blanches.

- Longueur de page : nombre maximum de lignes de texte sur une page. La longueur de page doit être comprise entre 0 et 66 lignes.

- Tabulateur : dispositif permettant des signes en colonne, en tableau et de faire de l'indentation automatique. Il y a 5 tabulateurs permis au maximum. Chaque tabulateur(i) doit être compris entre le maximum(0,tabulateur(i-1)) et 100.

- Fonte : un jeu de caractères d'une certaine forme (style) et d'une certaine taille est appelé fonte ou police de caractère. Dans le cas de notre éditeur,les différentes fontes possibles sont : messenger, grec, italique, gras, helvesan.

- Justification à gauche : cadrage du texte sur une marge gauche. Les deux solutions possibles sont : justifié à gauche ou non justifié à gauche.

- Justification à gauche et à droite : cadrage du texte sur une marge gauche et sur une marge droite. Les deux solutions possibles sont : justifié à gauche et à droite ou non justifé à gauche et à droite.

- Vérification de la césure : possibilité de changer la césure des mots. Les deux possibilites sont : vérifier la coupure des mots ou ne pas vérifier la coupure des mots.

- Nombre maximum de blancs entre deux mots : détermine l'espacement maximum entre deux mots. Ce espace doit être supérieur ou égal à 1.

Répertoire :

Un répertoire est un ensemble de textes d'un même utilisateur.

Utilisateur :

Un utilisateur est une personne ayant un répertoire sur l'ordinateur. Un utilisateur est caractérisé par son nom.

Session : Une session de l'éditeur est une occurrence d'exécution de l'éditeur.

Indentation : L'indentation est un mécanisme de décalage du texte vers la droite.

CR : Retour de chariot.

LF : Saut de ligne.

FF : Saut de page.

Chapitre 3

ANALYSE ORGANIQUE

Chapitre 3: Analyse organique.

(Françoise Fourny)

1. Introduction.

Dans ce chapitre, nous allons parler des choix d'implémentation qui ont été pris afin de réaliser toutes les spécifications décrites dans l'analyse fonctionnelle. Il nous a paru inutile d'insérer dans cette partie du mémoire l'analyse organique détaillée. Nous l'avons donc laissée en annexe (Voir annexe A).

Quels sont les aspects de l'analyse qui sont abordés dans ce chapitre ? Ils sont de trois types. Tout d'abord le choix du langage de programmation est évoqué dans une première partie. Dans ce paragraphe, nous faisons un bref historique du langage choisi et nous expliquons les principales caractéristiques de celui-ci.

Ensuite, une deuxième partie explique l'architecture choisie et donne les contraintes qui ont conduit à la construction d'une telle architecture.

Pour terminer, nous définissons le choix du support pour la mémorisation des documents, la manière de réaliser l'accès à ces documents et la structure du fichier document.

2. Le choix du langage.[3]

Le langage de programmation choisi est le langage C. Quelques détails sur les raisons de ce choix suivent.

2.1. Historique.

Le langage C fut au départ écrit et implémenté par Dennis Ritchie pour le système d'exploitation UNIX sur le DEC PDP-11.

C'est un langage dérivé du BCPL développé par Martin Richards. Bien que possédant certaines caractéristiques du BCPL, le C n'en est pas un dialecte.

Le C est un langage de programmation non-orienté. Ce langage comporte les caractéristiques suivantes : économie d'expressions, richesse des structures de contrôle, possibilité de structuration des données , grand jeu d'opérateurs et accès à un grand nombre de types d'objets. Ce n'est pas un langage de très haut niveau et il n'est pas spécialisé à un domaine particulier d'applications. Mais, l'absence de restrictions et sa généralité en font un moyen plus efficace et adéquat pour traiter beaucoup de tâches que certains langages supposés puissants. Cependant, cette absence de restrictions constitue un danger pour un programmeur non averti.

Le C n'est pas lié à un matériel particulier ou à un système. Il est relativement facile d'écrire des programmes qui vont tourner avec peu de changements sur chaque machine qui supporte le C.

Le C peut être appelé un "system programming language" car il est utile pour écrire le système d'exploitation. Il a aussi été utilisé pour écrire un grand nombre de systèmes de traitement numérique , systèmes de traitement de textes et également des programmes de gestion de Base de données. La plupart des programmes d'applications de UNIX(environ 90%), le système d'exploitation de UNIX et le compilateur C sont écrits en C. Il existe aussi des compilateurs C pour beaucoup d'autres machines, tel que l'IBM 370 , l'Honeywell 600 , l'Interdata 8/32 et des micro-processeurs tels que le Z80, le Z8000 et le M68000.

2.2. Présentation du langage C.

Le C est donc, comme il a été dit plus haut, un langage de programmation de relativement bas niveau. C'est à dire que le C traite les mêmes objets que la plupart des ordinateurs traitent: les caractères, les nombres, les adresses. Le C ne fournit pas d'opérations traitant directement les objets composés tels que les chaînes de caractères, les listes, les tables considérées comme un tout. Le C ne fournit pas de gestion dynamique de mémoire si ce n'est la pile fournie par les variables locales des fonctions.

Le C offre seulement quatre types de structures de contrôle :

- la séquence
- le test (if then else - switch)
- la boucle (while - for)
- le sous-programme fonction

2.3. Raisons du choix.

Les raisons qui ont conduit au choix du langage C comme outil d'implémentation sont multiples. Tout d'abord, nous savions que nous allions travailler sur un PDP-11 et sur un Smaky 8 sur lesquels tourne ou doit tourner le système UNIX. Le système d'exploitation de UNIX supporte le C et le compilateur C (version 6) sous UNIX est assez bon. De plus, les caractéristiques du C en font un outil de travail adéquat pour implémenter un système de traitement de textes. Enfin, le C est un langage connu à l'institut.

3. L'architecture du système.

Dans cette partie du chapitre concernant l'analyse organique, l'architecture du système analysé a été définie de façon complète. C'est à dire que autour des modules fonctionnels viennent se greffer des modules qui sont nécessaires à la réalisation du système mais qui du point de vue fonctionnel n'entrent pas en ligne de compte car ils ne présentent pas une unicité de traitement. Nous voulons parler ici des modules de gestion de fichiers,de gestion de l'écran, et du module d'erreur...

De plus, toutes les fonctions décrites dans l'analyse fonctionnelle ne doivent pas être considérées comme des modules séparés : il va falloir regrouper certaines fonctions ou bien encore en diviser d'autres. Cette redécoupe des traitememts est nécessaire afin d'avoir un meilleur agencement de ceux-ci.

La définition de l'architecture comportait une contrainte de modularité due au fait que le système devait être portable sur deux sites différents (voir chapitre 7). En effet, nous devions pouvoir implémenter sans trop de modifications le système sur un PDP 11/45 et sur un Smaky 8 . Ce problème de portabilité sera traité dans le chapitre 7. Une seconde raison a à la contrainte est le fait que le système devait être facile de maintenance.

3.1. Raisons du choix de l'architecture.

Précisons d'abord quelques caractéristiques que nous avons voulu donner à l'architecture de ce système.

La première de ces caractéristiques, nous l'avons déjà dit, est une très grande modularité. En effet, nous avons voulu faire un système où les modifications à apporter pour un problème sont localisées à un endroit du système et permettre ainsi une mise à jour plus facile des programmes. Cette facilité de modification est importante car le système a été développé sur un PDP-11 et doit être adapté à un Smaky 8. Ces 2 configurations sont fort différentes l'une de l'autre. Nous parlons des problèmes causés par la portabilité dans le chapitre 8. Une deuxième caractéristique est le fait qu'il fallait que ce soit implémentable graduellement.

Au vu de ces deux caractéristiques, nous avons donc choisi une architecture très découpée. Chaque module a une taille limitée et a une unité d'exécution c'est-à-dire que chaque module exécute une opération entièrement.

Une deuxième caractéristique de cette architecture est une découpe en niveau. Il y en a 6 qui deviennent de plus en plus complexes au fur et à mesure que l'on descend. Au premier niveau, nous avons donc un module qui permet de se diriger une première fois dans l'éditeur vers :

- l'impression
- l'édition
- le retour panique

- l'arrêt.

Ensuite au niveau 2, on retrouve les 4 modules correspondant aux 4 possibilités de choix du niveau 1.

Au niveau 3, sont situés tous les modules directement utilisés par le niveau 2, c'est-à-dire

- création
- modification
- lecture
- sauvetage
- destruction
- format
- activation
- panique
- quitter
- impression
- stop

Le niveau 4, est constitué par 2 modules :
-formatage
-gestion des primitives.

Le niveau 5 est formé de tous les modules relatifs aux primitives d'action sur le texte.

Le niveau 6, lui, est constitué des modules de gestion d'écran, de gestion du fichier, de gestion des erreurs et de modules divers.

3.2. Spécifications.

Les spécifications dégagées au cours de l'analyse organique étant assez volumineuses, celles-ci ont été laissées en annexe. Pour chaque module, nous avons défini les entrées, les sorties, les préconditions et les postconditions, la dynamique, la description sommaire, le traitement, le pseudo-code.

Schéma de l'architecture.

```
A initialisation
        B  affich_init (CALLED)
        B  erreur (CALLED)
        B  impression_ph_1 (CALLED by initialisation)
                C  affich_impr (CALLED)
                C  impression_fct (CALLED)
                        D  acquis-verif-valid-nom-doc (PART OF)
                        D  impression (PART OF)
                        D  erreur (CALLED)
                C  stop_fct_B (CALLED)
        B  edition_ph_C (CALLED by initialisation)
                C  affich_edit (CALLED)
                C  lecture_choix (CALLED)
                C  creation_fct_2 (CALLED)
                        D  affich_creat (CALLED)
                        D  acquis_verif_valid_nom_doc (PART OF)
                        D  get_fich (CALLED)
                                E  erreur (CALLED)
                                E  copie_fich (PART OF)
                        D  take_coord (CALLED)
                                E  erreur (CALLED)
                                E  mémoriser (PART OF)
                        D  création_entête (PART OF)
                        D  initialisation_format (PART OF)
                        D  formatage_fct (CALLED)
                                E  affich_fmt (CALLED)
                                E  acquis_vérif_val_fmt (PART OF)
                        D  gestion des primitives (CALLED)
                                E  lecture du choix (CALLED)
                                E  justification (CALLED)
                                E  pagination (CALLED)
                                E  vis.car.spe. (CALLED)
                                E  saut de page (CALLED)
                                E  caractère (CALLED)
                                E  insertion (CALLED)
                                E  DEL (CALLED)
                                E  BS (CALLED)
                                E  défaire (CALLED)
                                E  montre (CALLED)
                                E  copie (CALLED)
                                E  recherche (CALLED)
                                E  change (CALLED)
                                E  minuscule (CALLED)
                                E  majuscule (CALLED)
                                E  efface (CALLED)
                                E  depl_B_bas (CALLED)
                                E  depl_B_haut (CALLED)
                                E  depl_4_bas (CALLED)
                                E  depl_4_haut (CALLED)
                                E  depl_car_gau (CALLED)
                                E  depl_car_drt (CALLED)
                                E  depl_mot_gau (CALLED)
                                E  depl_mot_drt (CALLED)
                                E  depl_deb_txt (CALLED)
                                E  depl_fin_txt (CALLED)
```

```
                    E  centre (CALLED)
                    E  encadre (CALLED)
                    E  retour (PART OF)
                    E  erreur(CALLED)
                    E  gestion d'écran (CALLED)
                    E  gestion fichier (CALLED)
C  lecture_fct_2 (CALLED)
        D  acquis_verif_valid_nom_doc (PART OF)
        D  erreur (CALLED)
        D  get_fich (CALLED)
                E  erreur (CALLED)
                E  copie_fich (PART OF)
        D  take_coord (CALLED)
        D  depl_B_bas (CALLED)
        D  depl_B_haut (CALLED)
        D  depl_E_bas (CALLED)
        D  depl_E_haut (CALLED)
        D  depl_car_gau (CALLED)
        D  depl_car_drt (CALLED)
        D  depl_mot_gau (CALLED)
        D  depl_mot_drt (CALLED)
        D  depl_deb_txt (CALLED)
        D  depl_fin_txt (CALLED)
        D  stop (PART OF)
C  destruction_fct_2 (CALLED)
        D  affich_dest (CALLED)
        D  acquis_verif_valid_nom_doc (PART OF)
        D  erreur (CALLED)
        D  modif_flag_S_D (PART OF)
C  activation_fct_2 (CALLED)
        D  affich_act (CALLED)
        D  acquis_verif_valid_nom_doc (CALLED)
        D  erreur (CALLED)
        D  modif_flag_D_A (PART OF)
C  modification_fct_2 (CALLED)
        D  affich_modif (CALLED)
        D  acquis_verif_valid_nom_doc (PART OF)
        D  get_fich (CALLED)
                E  erreur (CALLED)
                E  copie_fich (PART OF)
        D  take_coord (CALLED)
                E  erreur (CALLED)
                E  mèmoriser (PART OF)
        D  mise_à_jour entète (CALLED)
        D  gestion des primitives (CALLED)
                E  lecture du choix (CALLED)
                E  justification (CALLED)
                E  pagination (CALLED)
                E  vis.car.spe. (CALLED)
                E  saut de page (CALLED)
                E  caractère (CALLED)
                E  insertion (CALLED)
                E  DEL (CALLED)
                E  BS (CALLED)
                E  défaire (CALLED)
                E  montre (CALLED)
                E  copie (CALLED)
```

```
                                E  recherche (CALLED)
                                E  change (CALLED)
                                E  minuscule (CALLED)
                                E  majuscule (CALLED)
                                E  efface (CALLED)
                                E  depl_B_bas (CALLED)
                                E  depl_B_haut (CALLED)
                                E  depl_4_bas (CALLED)
                                E  depl_4_haut (CALLED)
                                E  depl_car_gau (CALLED)
                                E  depl_car_drt (CALLED)
                                E  depl_mot_gau (CALLED)
                                E  depl_mot_drt (CALLED)
                                E  depl_deb_txt (CALLED)
                                E  depl_fin_txt (CALLED)
                                E  centre (CALLED)
                                E  encadre (CALLED)
                                E  retour (PART OF)
                                E  erreur(CALLED)
                                E  gestion d'écran (CALLED)
                                E  gestion fichier (CALLED)
                        C  sauvetage_fct_2 (CALLED)
                                D  affich_sauv (CALLED)
                                D  acquis_verif_valid_nom_doc (PART OF)
                                D  erreur (CALLED)
                                D  modif_flag_S_D (PART OF)
                                D  sauver_si_nom_different (PART OF)
                        C  panique_fct_2 (CALLED by edition_ph_2)
                                D  affich_pan (CALLED)
                                D  creat_pan (PART OF)
                                D  erreur (CALLED)
                                D  sauver_temporaire (PART OF)
                        C  format_fct_2 (CALLED by edition_ph_2)
                                D  affich_fmt (CALLED)
                                D  erreur (CALLED)
                                D  acquis_verif_valid_nom_doc (PART OF)
                                D  take_coord (CALLED)
                                D  formatage (CALLED)
                                        E affich_fmt (CALLED)
                                                F  fenetre (CALLED)
                                                F  rempl_ent (CALLED)
                                                F  rempl_fen (CALLED)
                                        E verif_acquis_val_fmt (PART OF)
                                        E erreur (CALLED)
                        C  quitter_fct_2 (CALLED by edition_ph_2)
                                D  sauver_fich (PART OF)
                                erreur (PART OF)
                B  retour_panique (CALLED by initialisation)
                        C  affich_panique (CALLED)
                        C  affich_document (PART OF)
                        C erreur (CALLED)
                B  stop (PART OF initialisation)
```

4. Structure du fichier.

Présentons le raisonnement qui a amené au choix de la structure du fichier.

Pour des raisons de sécurité, le système de traitement de texte ne travaille pas directement sur le fichier original. Le fichier original est recopié sur un fichier temporaire et l'état du fichier original n'est modifié que s'il y a un sauvetage explicite sur celui-ci.

Deux types d'accès sont possibles pour accéder au texte :
- l'accès séquentiel.
- l'accès direct.

Au vu des accès effectués par l'utilisateur dans un système de traitement de textes, il est nécessaire pour le stockage du fichier temporaire d'utiliser un support permettant un accès direct. Enumérons différents supports à accès direct :

- mémoire de masse
  - .disque
  - .floppy...
- mémoire centrale

Les caractéristiques de ces 2 types de supports à accès direct diffèrent.

Pour la mémoire de masse :

- taille importante
- permanence de l'information tant qu'elle n'est pas détruite
- obligation d'amener l'information en mémoire pour être traitée. Cette opération d'Entrée/Sortie est lente.

Pour la mémoire centrale :

- taille limitée
- mémoire volatile : si on coupe le courant, l'information est perdue.
- accès rapide aux informations

Après avoir choisi le type d'accès, il fallait encore trouver une solution au stockage du document sur ces supports. Plusieurs options étaient possibles :

- fichier temporaire totalement en mémoire centrale
- fichier temporaire partiellement en mémoire de masse

4.1. Fichier temporaire totalement en mémoire centrale.

La possibilité de stockage total du fichier temporaire en mémoire centrale, bien que la plus rapide, fut écartée pour deux raisons. Tout d'abord, le fait que la mémoire centrale soit volatile. Si l'alimentation de l'ordinateur était interrompue,

tout serait perdu. Ensuite, la place disponible par chaque utilisateur en mémoire centrale est limitée . Ceci pose un problème si le texte est trop grand.

4.2. Fichier temporaire partiellement en mémoire de masse.

Ce système est simple : une partie du fichier temporaire est sur mémoire de masse et une autre partie est dans la mémoire centrale.

La vitesse d'accès aux informations en mémoire centrale est très rapide. Mais lorsque l'utilisateur désire accéder à une ligne qui se trouve en mémoire de masse, il faut faire des Entrée/Sortie ce qui est beaucoup plus lent.

4.3. Solution retenue.

Dans le cas présent, cette solution de compromis est la meilleure car elle permet de meilleures performances; cette notion de performance est très importante dans un système de traitement de textes.

Une question se pose : comment choisir la taille de la zone tampon en mémoire centrale ? La partie du document qui se trouve en mémoire centrale à un moment donné est l'environnement de la ligne courante. Le choix de la taille de cet environnement a été simple: on stocke en mémoire centrale le nombre de lignes maximum que la fenêtre définie par l'utilisateur est capable de visualiser. Ensuite, chaque fois que l'utilisateur se déplace dans le texte, l'environnement glisse sous la fenêtre. Chaque ligne sortant de la fenêtre est recopiée à sa place dans le fichier.

Essayons maintenant de justifier le choix de la taille de la mémoire tampon. Il nous a semblé qu'utiliser une mémoire tampon de cette taille afin de mémoriser l'environnement courant était suffisant pour plusieurs raisons. Tout d'abord, le fait que ce choix permettait de faciliter la programmation; car aussitot que nous voulons faire référence à une ligne de texte qui ne se trouve pas affichée nous devons faire un accès disque alors que dans le cas où nous utilisons une mémoire tampon plus grande, il faut chaque fois se poser la question de savoir si nous devons faire un accès disque ou non. Ensuite, lorsque nous faisons une insertion d'une nouvelle ligne, si cette ligne se trouve au debut de la mémoire tampon, il faudra décaler deux fois plus de lignes dans le cas où nous choisissons un tampon double que dans le cas présent : ce décalage des lignes prend du temps alors qu'il faut que nous ayons de bonnes performances au point de vue rapidité. Enfin, chaque fois que nous devons remplir complètement cette mémoire tampon il faudra deux fois plus de temps si la mémoire tampon a une longeur deux fois plus grande. On peut dire qu'il y aura un assez grand nombre de remplissages totaux de la mémoire tampon, car quand l'utilisateur aura effectué une primitive sur le texte, il faut réafficher à l'écran une partie du fichier et donc remplir complètement la mémoire tampon.

Maintenant que nous avons choisi le support et la stratégie, il faut encore choisir le type d'organisation du fichier sur le support et le type d'accès de celui-ci.

Les possibilités de choix étaient séquentiel pur, direct, séquentiel indexé.

Le seul type d'organisation que le C permette est le séquentiel. En effet, il n'y a aucun moyen de spécifier que l'on désire travailler sur une autre organisation. Le choix du type d'organisation séquentielle est donc une obligation.

Le choix du type d'accès est évident. L'utilisateur doit à tout moment pouvoir se positionner n'importe où dans son fichier-document, donc on choisit l'accès direct.

Il en est de même que pour l'organisation en ce qui concerne le type d'accès car il n'y a aucune primitive d'accès direct en C, mais, il est possible grâce à une primitive spéciale (SEEK) de simuler l'accès direct en créant les primitives d'accès direct. Puisque ces primitives d'accès direct ont été crées, nous pouvons simplement dire que nous faisons de l'accès direct.

Il reste encore maintenant à définir la structure de ce fichier-document.

4.4. Structure du fichier.

Le fichier comporte 3 types d'enregistrement :

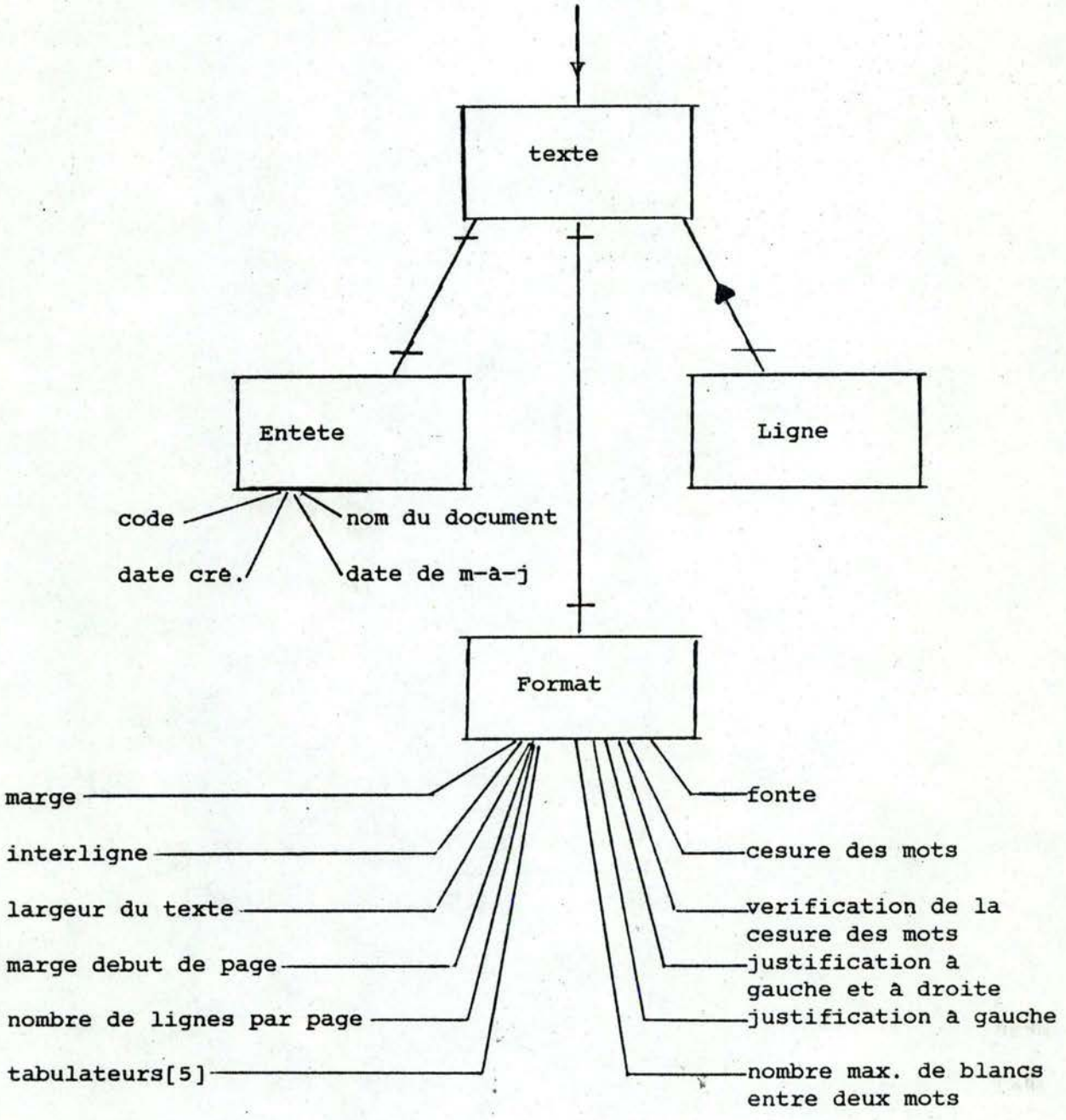
- un enregistrement FORMAT
- un enregistrement ENTETE
- des enregistrements ligne de texte.

Un fichier document comporte un et un seul enregistrement FORMAT, un et un seul enregistrement d'ENTETE et de 0 à n enregistrements lignes de texte.

Le fichier document est implanté en séquentiel pur mais peut être accédé par une simulation d'accès direct.

La longueur de chaque type d'enregistrement est de 100. Nous avons défini une longueur unique pour tous les types d'enregistrement afin d'uniformiser le fichier. Cette longueur a été choisie égale à 100 puisque c'est le nombre de caractères maximum qu'une feuille de papier A4 peut comporter en largeur.

Schèma du fichier.



La description du symbolisme utilisé est expliqué en [15]. Pour la signification des noms de données, il faut aller voir dans le dictionnaire des données (Voir chapitre 2).

Chapitre 4

IMPLEMENTATION

Chapitre 4: Implémentation

(Fourny Françoise)

1. Introduction

Dans le chapitre implémentation de ce mémoire, nous allons parler de la méthode d'implémentation utilisée afin de réaliser le système de traitement de textes analysé; ce chapitre traitera encore du planning suivi dans l'implémentation des modules faisant partie du système.

La partie méthode d'implémentation recouvre deux aspects : la méthode de programmation et la méthode de test. Dans une première partie de chapitre, la méthode de programmation est expliquée en se basant sur 2 caractéristiques. Ces deux caractéristiques sont la méthode de programmation utilisée et le problème des performances. La méthode de programmation choisie est la conception descendante.

Dans une deuxième partie de ce chapitre, la méthode de test de chaque module est expliquée. En fait, nous ne pouvons dire que le processus de test était une méthode approfondie de test mais une certaine structuration a été suivie.

La partie planning d'implémentation et des tests des modules expliquera l'étalage dans le temps de l'implémentation et du test des différents modules et donnera le pourquoi de cette répartition.

2. Méthode de programmation .

Dans notre façon de procéder, la méthode de programmation et la méthode de test sont assez liées. En effet, nous avons mené la programmation et les tests de façon progressive et conjointe. Cette manière de procéder rend floue la limite entre la méthode de test et la méthode de programmation. Deux caractéristiques sont à dégager de notre méthode de programmation. La première de ces caractéristiques consiste à faire le choix de la méthode à utiliser pour l'implémentation. Deux méthodes sont possibles : TOP DOWN (descendante) et BOTTOM UP (ascendante).

La méthode TOP DOWN (descendante) est une méthode consistant à procéder par étapes en partant du niveau de la formulation du problème jusqu'au niveau de détail. A chaque étape, on affine les divers éléments de programme obtenus auparavant en les reprenant en termes d'opérations plus simples.

En opposition à cela, il y a la méthode BOTTOM UP. La méthode BOTTOM UP consiste à construire peu à peu des programmes de plus en plus complexes à partir du niveau le plus bas et en remontant vers le problème à résoudre. Il s'agit donc ici de fabriquer des outils avant d'aborder le problème qu'il faut résoudre.

Il est à noter que ces deux méthodes ne sont jamais utilisées comme telles car elles représentent deux extrêmes. Dans la pratique, elles sont utilisées ensemble par un mélange adéquat de l'une et l'autre selon le problème posé.

La méthode de programmation utilisée en priorité dans notre cas est la méthode TOP DOWN. Considérant l'éditeur comme un arbre à 6 niveaux il fut aisé d'employer cette méthode.

Expliquons le cheminement suivi : nous avons procédé par étapes . Dans une première étape : nous sommes partis du premier niveau. Nous avons écrit le module constituant le niveau 1 (le module initialisation) et chaque fois qu'il y avait appel à une prodédure d'un autre niveau, nous avons simulé le traitement de cette procédure. C'est-à-dire que tous les modules appelés par le niveau 1 ont été écrits mais le traitement à effectuer était remplacé par l'écriture du libellé du module. Une fois que ce module du niveau 1 marchait, nous sommes alors passés à la deuxième étape de programmation et de test : tous les modules appelés par le niveau 1 ont été écrits et les modules appelés par ceux-ci ont été simulés de la même façon qu'à l'étape 1.

Tout en avançant de cette façon par étape, nous sommes arrivés au niveau 5 où nous n'avons pu continuer comme cela. A partir de ce moment, nous faisions appel à des modules qui devait avoir un effet immédiat sur le module appelant : nous voulons parler par là des modules d'accès direct (read-rand, write-rand...). Nous avons donc écrit ces modules et testé ceux-ci en dehors de l'éditeur dans de petits programmes de test. Un fois ces modules au point, nous avons repris le test selon la méthode TOP DOWN.

Pourquoi ce changement de méthode ? Nous avons effectué des tests sur les modules du niveau 5 après avoir testé les primitives d'accès direct car il nous a semblé qu'il serait plus facile de tester celui-

ci si nous étions sûrs que les primitives d'E/S en accès direct faisaient bien ce qu'elles étaient supposées faire. Nous pouvons ainsi les utiliser comme des primitives qui seraient offertes par le langage.

La deuxième caractéristique de notre méthode de programmation est le fait que nous n'avons pas tenu compte des problèmes de performances. Il nous semble impossible de pouvoir répondre en même temps aux exigences de performance et de fiabilité d'un programme. Comment pouvoir dire qu'un programme a une grande vitesse si l'on n'est pas sûr de ce que l'on fait ? Etant donné ce choix de ne pas tenir compte des performances, les modules écrits l'ont été de façon à ce que cela marche mais pas à ce que cela marche avec de bonnes performances.

3. Compléments relatifs aux tests.

Rappelons que la méthode de test utilisée pour chaque module n'est pas très élaborée. Elle suit de très près la méthode de programmation dans la façon de tester l'utilisation des modules appelés au différents niveaux.

Deux caractéristiques apparaissent dans notre méthode de tests. La première consiste en trois critères de test d'un module. Ces trois critères sont :

- Chaque instruction est exécutée au moins une fois.
- Chaque décision du module est évaluée au moins un fois avec l'issue vraie, et au moins une fois avec l'issue fausse.
- Chaque condition élémentaire dans chaque décision est évaluée au moins une fois avec un valeur vraie, et au moins une fois avec la valeur fausse.

La seconde des caractéristiques de test d'un module se trouve dans la conception des jeux de test. Dans la façon de tester chaque module, nous avons utilisé des jeux de test avec des difficultés croissantes. Nous avons commencé par utiliser des jeux de test assez généraux et ensuite quand cela marchait nous recommencions à tester avec un jeu de test dont la complexité était plus grande.

4. Planning des tests.

Après avoir terminé la partie analyse de ce mémoire, un premier planning de test avait été élaboré. Ce planning prévoyait que les niveaux 1, 2, 3, 4 seraient terminés complètement pour la fin du mois de décembre. De cette façon, il nous restait quatre mois afin de tester le niveau 5 et le niveau 6. Mais le programme des tests a été fortement retardé à cause des difficultés que nous avons rencontrées en stage. Ces difficultés étaient de trois ordres :

- l'apprentissage du langage en même temps que la programmation.
- la valeur du compilateur C.
- le fait que nous étions seuls à connaître le C à l'EPFL.

La première de ces difficultés est bien compréhensible si l'on connait la complexité du C.

La deuxième de ces difficultés est le compilateur C. La documentation accompagnant le compilateur C présentait des imprécisions et même des erreurs.

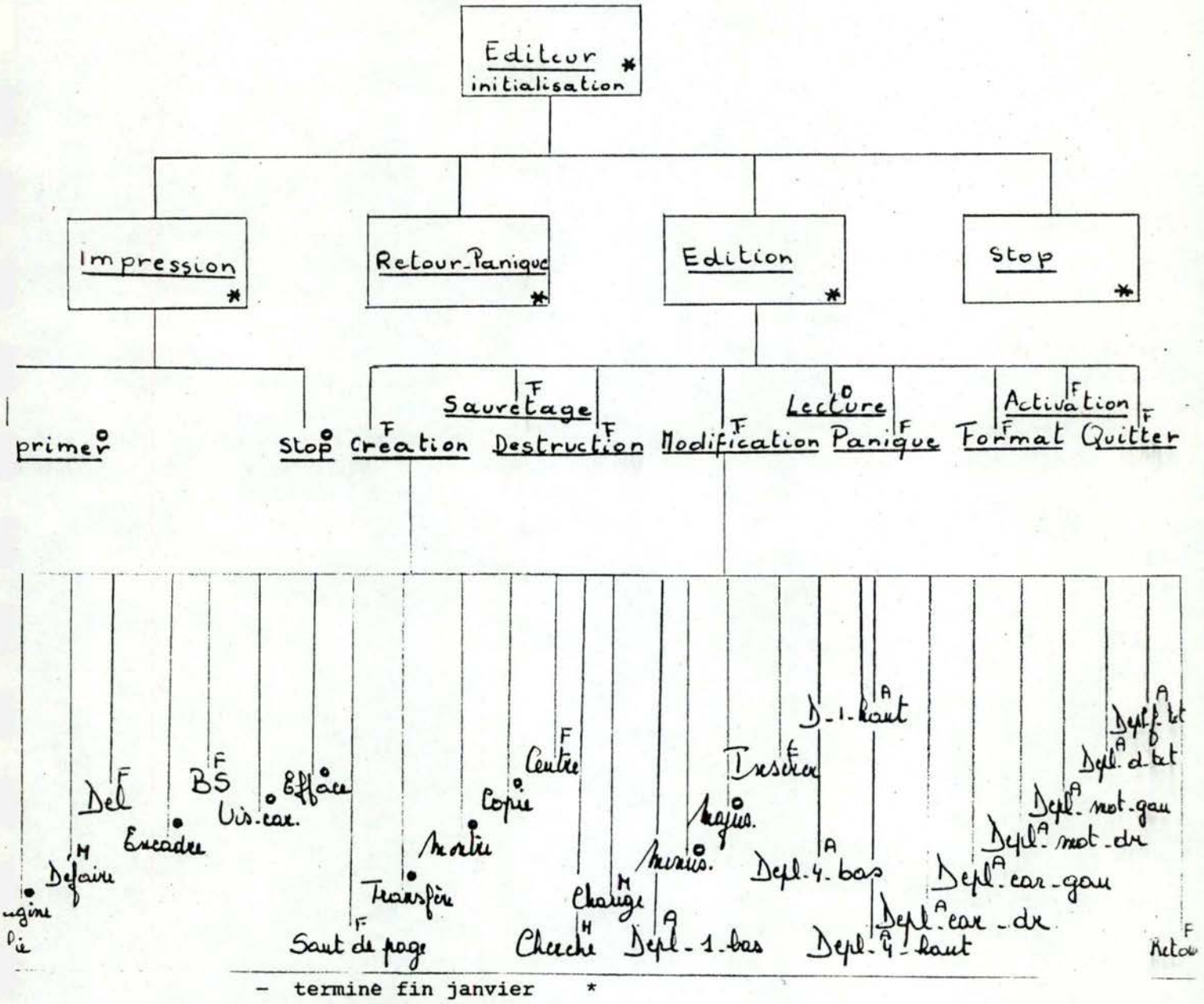
- Des exemples de programmes donnés entrés tel quel ne marchaient pas.
- Une autre partie des instructions ne marchait pas (Remove, &).
- Nous devions vider le buffer disque nous-même.
- Il n'y avait pas de possibilité d'appels système car ceux-ci sont construits avec le caractère $ qui est refusé par le compilateur.
- Un inconvénient important était l'impossibilité d'avoir un listing d'erreurs de compilation.

La troisième de ces difficultés est aussi très compréhensible. Quand un problème se posait nous ne pouvions demander de l'aide à personne car personne ne connaissait le C.

A cause de tout cela, quand le mois de décembre s'est terminé nous avions testé les niveaux de 1 à 3. Ces niveaux étaient testés, mais non terminés car nous n'étions pas sûrs du compilateur. Une fois rentrés en Belgique, il nous a fallu rédiger un nouveau planning de tests qui fut cette fois respecté. Voici ce planning de tests :

- test des niveaux 1-2-3-4 terminé fin janvier
- test des primitives d'actions sur le texte (niveau 5) et de la gestion d'écran et gestion des fichiers réparti sur le reste du temps. Voir schéma du planning.

Schèma : planning des tests.



- terminé fin janvier *
- non implémenté O
- terminé fin février F
- terminé fin mars M
- terminé fin avril A

Parmi les routines, nous avons commencé à implémenter celles qui nous semblaient les plus importantes. Dans les routines que nous n'avons pas implémentées, certaines ne l'ont pas été car elles nécessitent un écran totalement graphique et le terminal utilisé pour l'implémentation est semi-graphique. C'est le cas pour encadre, caractère.

Chapitre 5

MANUEL D'UTILISATION

Chapitre 5: Manuel d'utilisation

(Jean-Marc Cheu)

1. Début de la session

Une session de l'éditeur commence en donnant l'ordre "edef" qui est l'appel de la version exécutable de l'éditeur. Cette version se trouve dans le directoire "/mnt/mem/fourny". L'utilisateur ne doit pas donner de paramètres après cet ordre. Une fois la commande d'appel de l'éditeur lancée, un premier menu va apparaître.

```
EDITEUR

                        M E N U
                        =======

                    _ E(dition)

                    _ I(mpression)

                    _ R(etour-panique)

                    _ S(top)


                    VOTRE CHOIX ? : _
```

A ce moment, l'utilisateur choisit une des options proposées en tapant l'initiale de l'option sélectionnée. Celle-ci peut être tapée en majuscule ou en minuscule. Elle est suivie de ‹RETURN›.

Les options proposées sont :

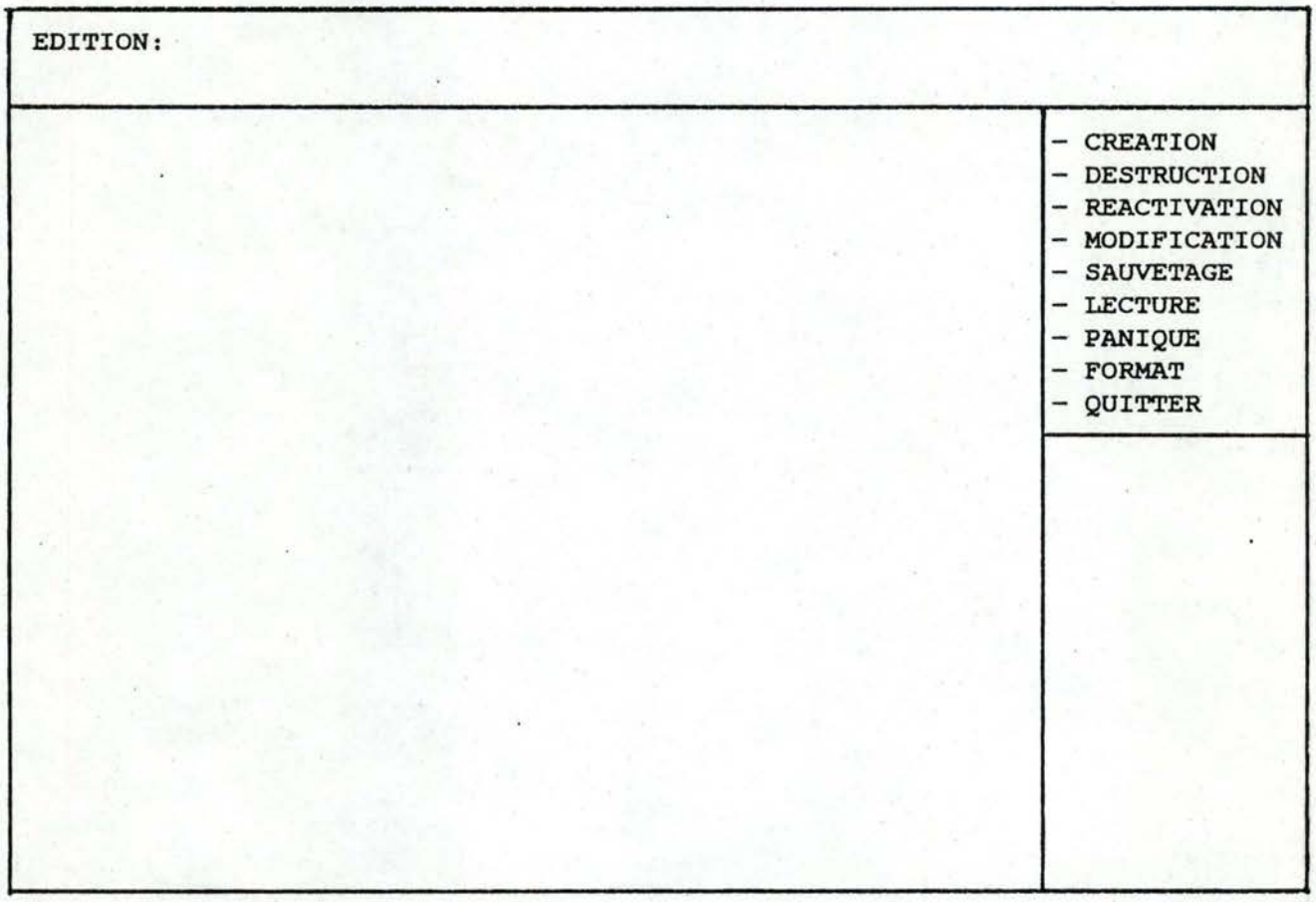
- E(dition)
- I(mpression)
- R(etour-panique)
- S(top)

En cas d'erreur, le message "CHOIX ERRONE! VEUILLEZ RECOMMENCER " apparaît dans le haut de l'écran et l'utilisateur doit recommencer son choix.

2. L'édition

2.1. Choix d'une action

Ayant choisi l'option d'édition, l'utilisateur voit apparaître l'écran suivant:



On y trouve un menu complémentaire permettant à l'utilisateur d'effectuer des actions d'édition. Ces actions sont :

- CREATION : pour créer un nouveau document
- DESTRUCTION : pour détruire un document existant
- REACTIVATION : pour restaurer un document qu'on avait détruit
- MODIFICATION : pour modifier un document existant
- SAUVETAGE : pour sauver les modifications apportées à un document
- LECTURE : pour voir le contenu d'un document
- PANIQUE : pour sortir en vitesse de l'éditeur

- FORMAT : pour voir et modifier si nécessaire les caractéristiques de formatage d'un document

- QUITTER : pour sortir du mode d'édition

Le curseur est alors automatiquement placé au début de ce menu. Pour se déplacer dans le menu, l'utilisateur doit employer les touches de déplacement vertical du curseur (cfr paragraphe 2.2).

Ces touches sont :

- CTRL / c
- CTRL / x
- CTRL / v
- CTRL / r
- CTRL / e
- CTRL / t

En cas d'erreur, le curseur ne bouge pas de place. Une fois arrivé devant l'action choisie, l'utilisateur n'a plus qu'à pousser sur la touche ‹RETURN›.

2.2. Le déplacement du curseur

- CTRL / c : déplacement d'une ligne vers le bas

- CTRL / x : déplacement de quatre lignes vers le bas

- CTRL / v : déplacement à la fin du texte pour l'insertion et la modification, à la fin de l'écran pour la détermination des coordonnées de la fenêtre et à la fin du menu pour le choix dans ce menu.

- CTRL / t : déplacement au début du texte pour l'insertion et la modification, au début de l'écran pour la détermination des coordonnées de la fenêtre et au début du menu pour le choix dans ce menu.

- CTRL / e : déplacement de quatre lignes vers le haut

- CTRL / r : déplacement de une ligne vers le haut

- CTRL / s : déplacement de un mot vers la gauche pour l'insertion et la modification et de dix caractères vers la gauche pour la détermination des coordonnées de la fenêtre.

- CTRL / g : déplacement de un mot vers la droite pour l'insertion et la modification et de dix caractères vers la droite pour la détermination des coordonnées de la fenêtre.

- CTRL / d : déplacement de un caractère vers la gauche

- CTRL / f : déplacement de un caractère vers la droite

Pour cela, il faut en même temps appuyer sur la touche CTRL et sur la lettre voulue pour effectuer le déplacement. Cette lettre peut être tapée en minuscule ou en majuscule.

2.3. L'action "création"

La fonction de création permet de créer un document qui n'existe pas encore.

2.3.1. Détermination du nom du document

Un nouvel écran apparaît où le précédent menu a été effacé et a été remplacé par l'écran ci-dessous :

EDITION : NOM DU DOCUMENT :	
	- CREATION
	- Justifie - Pagine - Vis.car.spec. - Saut de page - Montre - Copie - Recherche - Change - minuscule(m) - Majuscule(M) - Transfère - Caractère - Defait - Encadre - Efface - Centre - Retour

L'éditeur imprime à l'écran le libellé "NOM DU DOCUMENT : " Le curseur est alors placé à la droite de ce libellé.

Si l'utilisateur est entré par erreur dans cette fonction, il lui suffit, pour en sortir, de taper directement sur la touche <RETURN>.

Le nom du document ne doit en aucun cas dépasser la longueur de 14 caractères au total. Il est composé du nom proprement dit du document (max 10 caractères) , d'un point (facultatif) et de son extension (max 3 caractères et facultatif) qui varie selon le contenu de ce document.

Si le nom du document est trop long, le message "LE NOM DU DOCUMENT EST TROP LONG " apparaîtra et l'utilisateur doit redonner un nouveau nom. Si l'utilisateur se trompe lors de la composition du nom, il peut employer la touche ‹delete› pour détruire le caractère indésirable. Quand le nom est composé, l'utilisateur n'a plus qu'à pousser la touche ‹RETURN› pour indiquer qu'il a fini. Si le nom du document existe déjà, le message " CE DOCUMENT EXISTE DEJA " apparaît à l'écran et l'utilisateur doit redonner un nouveau nom.

A ce moment là, l'utilisateur doit déterminer la fenêtre c'est-à-dire la zone de l'écran où il veut créer son document. Cette fenêtre réduit la taille de l'écran où l'utilisateur peut voir le document mais en même temps elle permet de voir plusieurs documents simultanément.

2.3.2. Détermination des coordonnées de la fenêtre

Un premier message apparaît au dessus de l'écran. Ce message est " COIN SUPERIEUR GAUCHE DE LA FENETRE PUIS ‹RETURN› ". Il demande à l'utilisateur de déterminer les coordonnées du coin supérieur gauche de la fenêtre puis de taper sur la touche ‹RETURN›

Un deuxième message " COIN INFERIEUR DROIT DE LA FENETRE PUIS ‹RETURN› " apparaît alors pour demander à l'utilisateur de déterminer les coordonnées du coin inférieur droit de la fenêtre puis de taper sur la touche ‹RETURN›

La détermination des coordonnées se fait au moyen des touches de déplacement du curseur. Si l'utilisateur tape une autre touche que celles du déplacement du curseur (à l'exception de la touche ‹RETURN›), le curseur restera à sa place.

La fenêtre doit obligatoirement avoir plus de 36 caractères de large et plus de 15 lignes de long. En cas d'erreur, le message approprié apparaît à l'écran et l'utilisateur doit recommencer tout le processus. Ces messages sont :

- la longueur de la fenêtre doit être › que 15 lignes
- la largeur de la fenêtre doit être › que 36 caractères
- erreur dans les coordonnées de la fenêtre

2.3.3. Détermination des caractéristiques de formatage du document

Une fois que l'utilisateur a déterminé les coordonnées de la fenêtre, celle-ci se dessine sur l'écran sous la forme suivante :

DOCUMENT : ‹nom du document› Date Creat : ‹dd-mmm-yy hh:mm› Date m-a-j | ‹dd-mmm-yy hh:mm›	code

L'entête est alors remplie par le programme avec le nom du document, la date de création du document sous la forme ‹dd-mmm-yy hh:mm› (dd: le jour, mmm: le mois sous la forme des 3 premières lettres du mois, yy: l'année, hh: l'heure et mm: les minutes) et la date de mise à jour du document. La date de mise à jour du document est la date à laquelle le fichier a été modifié pour la dernière fois.

Ensuite des caractéristiques de formatage vont s'inscrire dans cette fenêtre. L'utilisateur peut alors les accepter globalement (taper sur la touche ‹RETURN›) ou vouloir les modifier.

Ces caractèristiques apparaissent sous la forme suivante avec des valeurs par défauts :

```
EDITION :  NOM DU DOCUMENT :

                                                          - FORMAT

  DOCUMENT : <            > Date Creat. : <            >
                              Date m-a-j. : <            >

  Marge gauche                          :  10
  Largeur du texte                      :  80
  Interligne                            :   1
  Marge haut                            :   5
  Fin de page                           :  64
  Tabulateurs                           :   0   0   0   0   0
  Fonte                                 : MESSENGER
  Justification à gauche                : OUI
  Justification à gauche et a droite    : NON
  Coupure des mots                      : NON
  Verification des coupures             : NON
  Nombre maximum blancs entre 2 mots    :   2
```

Le curseur est placé sur la première valeur de ces caractèristiques. Pour passer d'une caractèristique à l'autre, il faut pousser successivement la touche <TAB> puis la touche <RETURN>.

Les différentes valeurs admissibles sont :

- marge gauche : 0 < valeur < 101
- largeur du texte : 0 < valeur < 100
- interligne : 0 < valeur < 5
- marge du haut : 0 < valeur < 67
- fin de page : 0 < valeur < 66
- tabulateurs : voir plus bas
- fonte : messenger,italique,helvesan,grec
- justif gauche : oui,non
- justif gau/droit : oui,non
- coupure des mots : oui,non
- verif coupure : oui,non
- nbre blancs ent 2 mots: 0 < valeur < 11

Les caractères peuvent être tapés en majuscules ou en minuscules. Ces caractères sont : messenger, italique, helvesan, grec, oui et non.

Chaque valeur numérique est un nombre de trois chiffres et doit être cadrée à droite. Prenons l'exemple de l'introduction du nombre 54 : le premier chiffre sera au

choix 0 ou un blanc, le deuxième sera 5 et le troisième sera 4.

Pour les tabulateurs, chaque valeur peut varier entre 0 et 100 mais elle doit être plus grande que la précédente. Pour passer d'une valeur de tabulateur à l'autre, il suffit de taper sur la touche d'espacement puis sur la touche <RETURN>. Pour sortir, à n'importe quel moment, de la série des tabulateurs, l'utilisateur doit pousser sur la touche <TAB> puis sur la touche <RETURN>.

Lors de la modification des valeurs de tabulateurs, après chaque introduction de valeur, si l'utilisateur pousse sur la touche <RETURN>, il passera à la valeur suivante de tabulation.

Pour terminer la détermination des caractéristiques de formatage, il suffit alors de pousser sur la touche <RETURN>.

A ce moment, la fenêtre se vide et , le curseur étant placé en haut et à gauche de la fenêtre, l'utilisateur peut maintenant introduire son texte.

2.3.4. Le texte

En plus des touches du clavier permettant d'introduire son texte, l'utilisateur dispose des touches de déplacement du curseur pour voyager dans la fenêtre comme il le désire (cfr paragraphe 2.2). Pour faire défiler le texte sur l'écran, l'utilisateur doit déplacer le curseur jusqu'au bord supérieur ou inférieur de la fenêtre. A ce moment, en continuant de pousser sur les touches de déplacement du curseur , l'utilisateur verra défiler le texte selon les touches poussées. Il dispose aussi d'un menu d'actions qu'il peut effectuer sur son texte. Ces actions sont :

- Justifie: justifie la totalité du texte selon les caractéristiques définies dans le format.

- Pagine: ajoute un numéro de page au bas de chaque page.

- Visualisation des caractères spéciaux : visualise les caractères TAB (caractère de tabulation), CR (retour de chariot), EOL (caractère de fin de ligne), FF (caractère de saut de page), SAJEX (caractère pseudo-blanc rajouté par la routine de justification), TIRET (caractère pseudo-tiret rajouté par la routine de justification pour la césure de mots), CADRATIN (caractère pseudo-blanc rajouté par la routine de justification pour éviter la coupure de mots). (*)

- Saut de page: force un saut de page à l'endroit où se trouve le curseur.

- Montre : montre les contenus de différents buffers. Ces buffers sont des zones temporaires (le temps d'une session de l'éditeur) de mémoire où l'utilisateur peut copier un texte ou une portion d'un texte jusqu'à concurrence de 20.000 caractères. (*)

- Copie : permet de copier une portion de texte appartenant à un document soit dans ce même document soit dans un autre. (*)

- Caractère : permet de définir un autre style de caractère que la fonte normale pour une partie du texte. (*)

- Recherche: recherche un string donné dans tout le texte. Cette recherche se fait dans tout le texte et elle s'arrête momentanément à la première occurence du string donné. Si l'utilisateur veut continuer la recherche sur le même string, il doit taper sur les touches CTRL / g et la recherche va continuer. Si l'utilisateur veut arrêter définitivement la recherche, il doit taper sur la touche ‹RETURN›.

- Change: recherche un string donné dans tout le texte et permet de le changer par un autre. Le principe est le même que pour la recherche d'un string. La seule différence est que, lorsque que l'on a trouvé une occurence du string donné, l'utilisateur décide à ce moment-là s'il désire changer ou non. S'il veut changer le string donné par un autre, il doit taper sur la touche ‹RETURN›. S'il ne veut pas changer cette occurence-là du string donné, il n'a qu'à taper sur les touches CTRL / g et la recherche va continuer jusqu'à la prochaine occurence du string donné.

- Minuscule: permet de changer une portion de texte délimitée par l'utilisateur en minuscule. (*)

- Majuscule: permet de changer une portion de texte délimitée par l'utilisateur en majuscule. (*)

- Transfère : effectue le transfert d'une partie du texte délimité par l'utilisateur à un endroit du texte défini aussi par l'utilisateur. (*)

- Défait: annule l'effet de la commande précédente au niveau des modifications dans un texte.

- Encadre : dessine un encadrement autour d'une partie du texte. (*)

- Efface : efface une partie du texte délimitée par l'utilisateur. (*)

- Retour: fin de création ou modification.

(*) : Ces actions ne sont pas encore implémentées.

Pour atteindre ce menu, l'utilisateur doit taper les touches CTRL / b. Il est alors placé au début du menu et dispose des touches de déplacement du curseur pour se déplacer dans le menu.

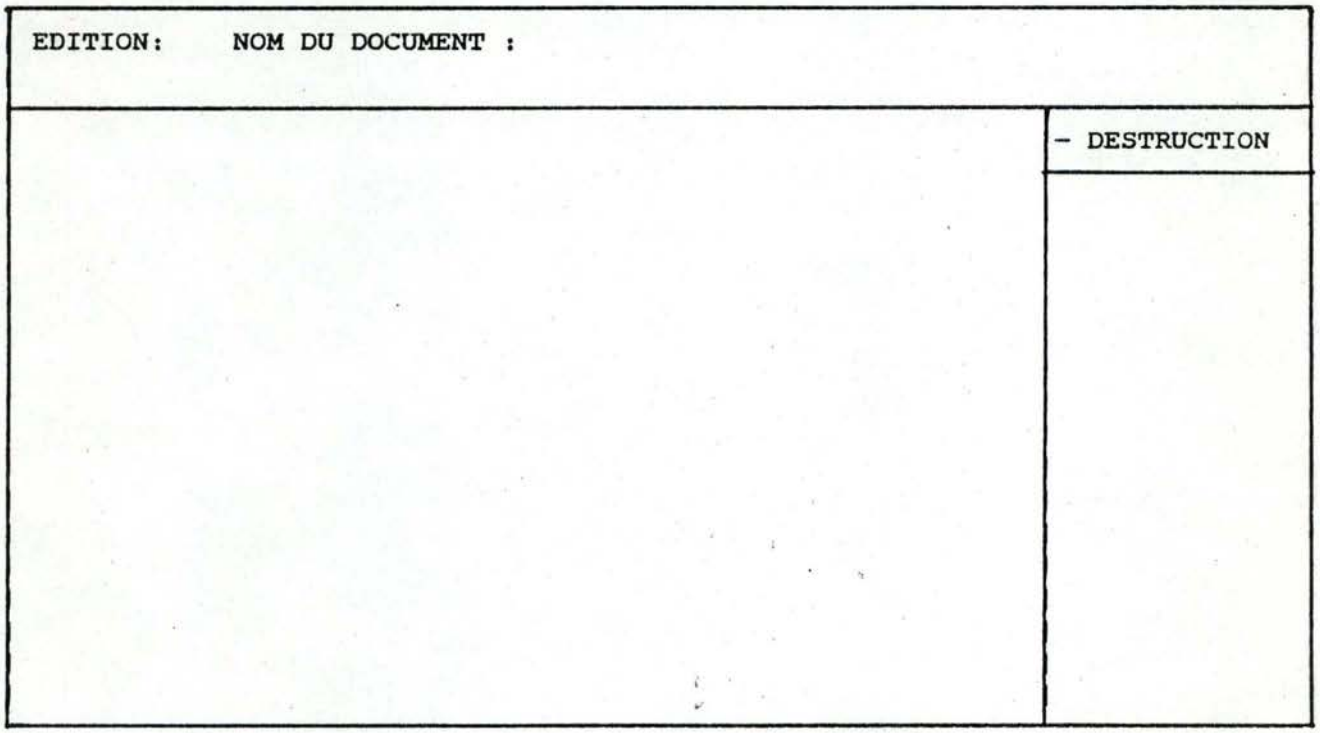
En cas d'erreur, le curseur ne bouge pas.

Une fois arrivé devant la fonction choisie, il suffit de pousser sur la touche <RETURN>. Pour revenir là où il était dans le texte, l'utilisateur n'a qu'à taper une nouvelle fois sur les touches CTRL / b.

2.4. L'action "destruction"

L'option DESTRUCTION conduit à la destruction d'un document dont on donne le nom. Cette destruction est logique jusqu'au moment où on donne l'ordre de "QUITTER" (fin d'édition). A ce moment-là, la destruction est physique.

Un nouvel écran apparaît où le précédent menu a été effacé et a été remplacé par l'écran ci-dessous :



Si l'utilisateur est entré par erreur dans la fonction, il lui suffit ,pour en sortir, de taper directement sur la touche <RETURN>.

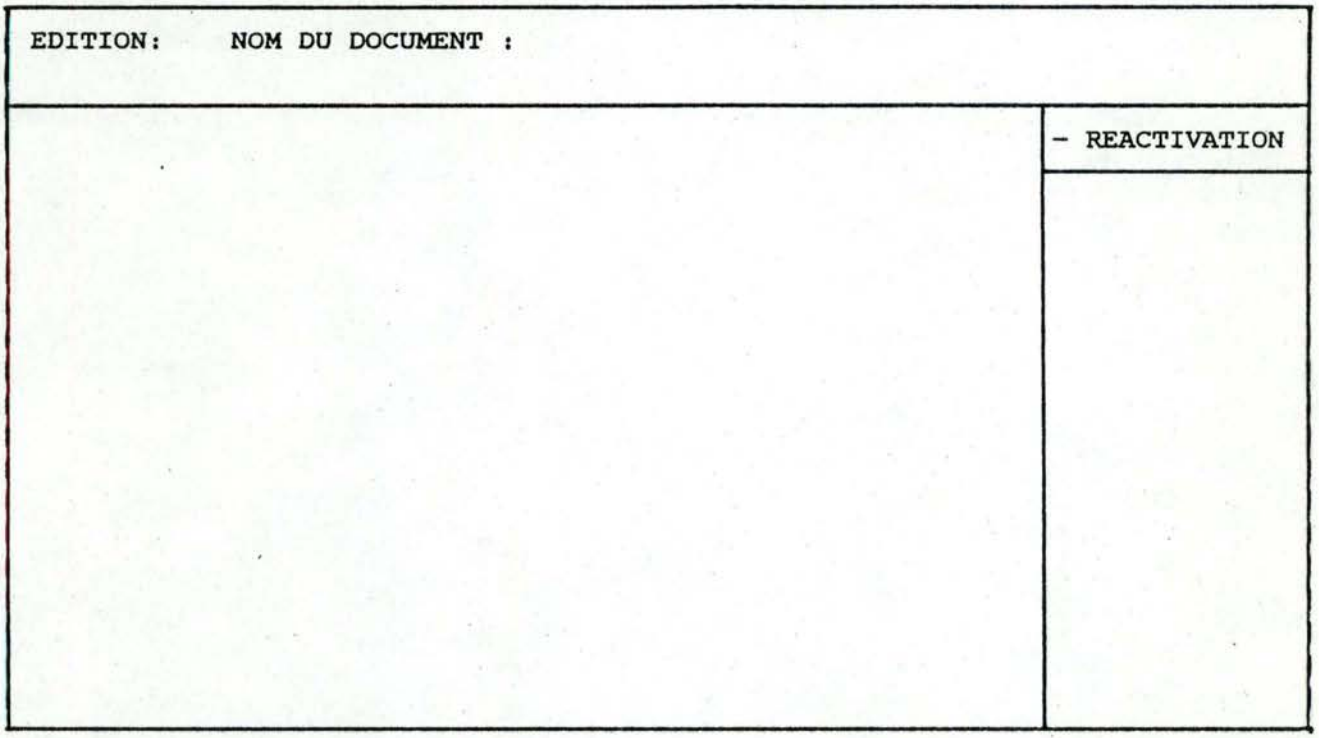
L'utilisateur indique le nom du document qu'il veut détruire (voir paragraphe 2.3.1). Le curseur est placé automatiquement à droite du libellé "NOM DU DOCUMENT".

En cas d'erreur, le message "IMPOSSIBLE DE DETRUIRE CE DOCUMENT CAR NON EDITE " apparaît et l'utilisateur doit resélectionner l'action de destruction.

2.5. L'action "reactivation"

L'option REACTIVATION est une opération qui a pour effet d'annuler l'opération de destruction d'un document.

Un nouvel écran apparaît où le précédent menu a été effacé et a été remplacé par l'écran ci-dessous :



Si l'utilisateur est entré par erreur dans la fonction, il lui suffit ,pour en sortir, de taper directement sur la touche <RETURN>.

Ensuite, l'utilisateur indique le nom du document qu'il veut activer (voir paragraphe 2.3.1). Le curseur est placé automatiquement à droite du libellé "NOM DU DOCUMENT".

En cas d'erreur, le message "ERREUR DE REACTIVATION DU DOCUMENT " apparaît et l'utilisateur doit resélectionner l'action d'activation.

2.6. L'action "modification"

L'option MODIFICATION est l'opération qui permet la mise à jour d'un document.

Un nouvel écran apparaît où le précédent menu a été effacé et a été remplacé par l'écran ci-dessous :

EDITION : NOM DU DOCUMENT :	
	- MODIFICATION
	- Justifie - Pagine - Vis.car.spec. - Saut de page - Montre - Copie - Recherche - Change - minuscule(m) - Majuscule(M) - Transfère - Caractère - Defait - Encadre - Efface - Centre - Retour

Si l'utilisateur est entré par erreur dans la fonction, il lui suffit ,pour en sortir, de taper directement sur la touche <RETURN>.

L'utilisateur indique le nom du document sur lequel il veut travailler (voir paragraphe 2.3.1). Le curseur est placé automatiquement à droite du libellé "NOM DU DOCUMENT".

Ensuite, il décide des coordonnées de la fenêtre dans laquelle il désire qu'apparaisse le document choisi (voir paragraphe 2.3.2).

A ce moment, le début du document apparaît dans la fenêtre et le curseur est placé au début du texte. L'utilisateur dispose de tous les moyens mis à sa disposition pour agir sur le texte (voir paragraphe 2.3.4).

2.7. L'action "sauvetage"

L'option de SAUVETAGE effectue le sauvetage du document dont l'utilisateur indique le nom. Le sauvetage est logique et n'est effectif que lors de la sortie de la phase d'édition.

Un nouvel écran apparaît où le précédent menu a été effacé et a été remplacé par l'écran ci-dessous :

EDITION: NOM DU DOCUMENT :	
	- SAUVETAGE

Si l'utilisateur est entré par erreur dans la fonction, il lui suffit ,pour en sortir, de taper directement sur la touche <RETURN>.

L'utilisateur indique le nom du document sur lequel il veut travailler (voir paragraphe 2.3.1). Le curseur est placé automatiquement à droite du libellé "NOM DU DOCUMENT".

En cas d'erreur, deux messages peuvent apparaître :

- si le document n'existe pas, le message "CE DOCUMENT N'EXISTE PAS" va apparaître
- si le document n'a pas été édité dans cette session de l'éditeur, le message "IMPOSSIBLE DE SAUVER CE DOCUMENT CAR NON EDITE " va apparaître.

Ensuite, l'utilisateur devra resélectionner l'action d'activation.

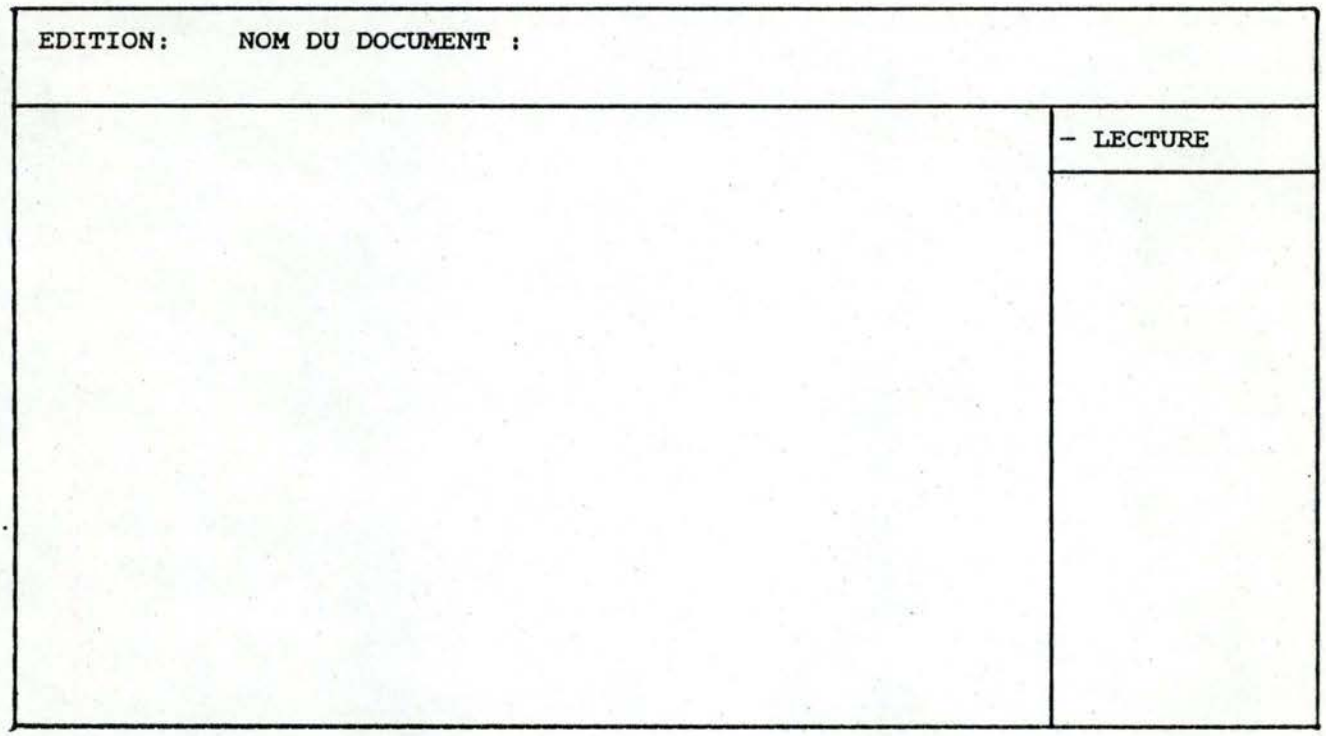
Il y a possibilité de sauver le document sous un nom différent de celui sous lequel le fichier a été manipulé dans la

session de l'éditeur. Pour cela, l'éditeur demande à l'utilisateur s'il désire sauver le document sous son premier nom (en tapant sur la touche <RETURN>) ou sous un autre nom (voir paragraphe 2.3.1) par le message " VOULEZ-VOUS SAUVER LE DOCUMENT SOUS UN AUTRE NOM ? SI NON TAPER <RETURN> SI OUI TAPER LE NOUVEAU NOM ".

2.8. L'action "lecture" (non implémentée)

L'option LECTURE permet de voir le contenu d'un document.

Un nouvel écran apparaît où le précédent menu a été effacé et a été remplacé par l'écran ci-dessous :



L'utilisateur indique le nom du document qu'il désire lire. Le curseur est placé automatiquement à droite du libellé "NOM DU DOCUMENT".

Si l'utilisateur est entré par erreur dans la fonction, il lui suffit, pour en sortir, de taper directement sur la touche <RETURN>.

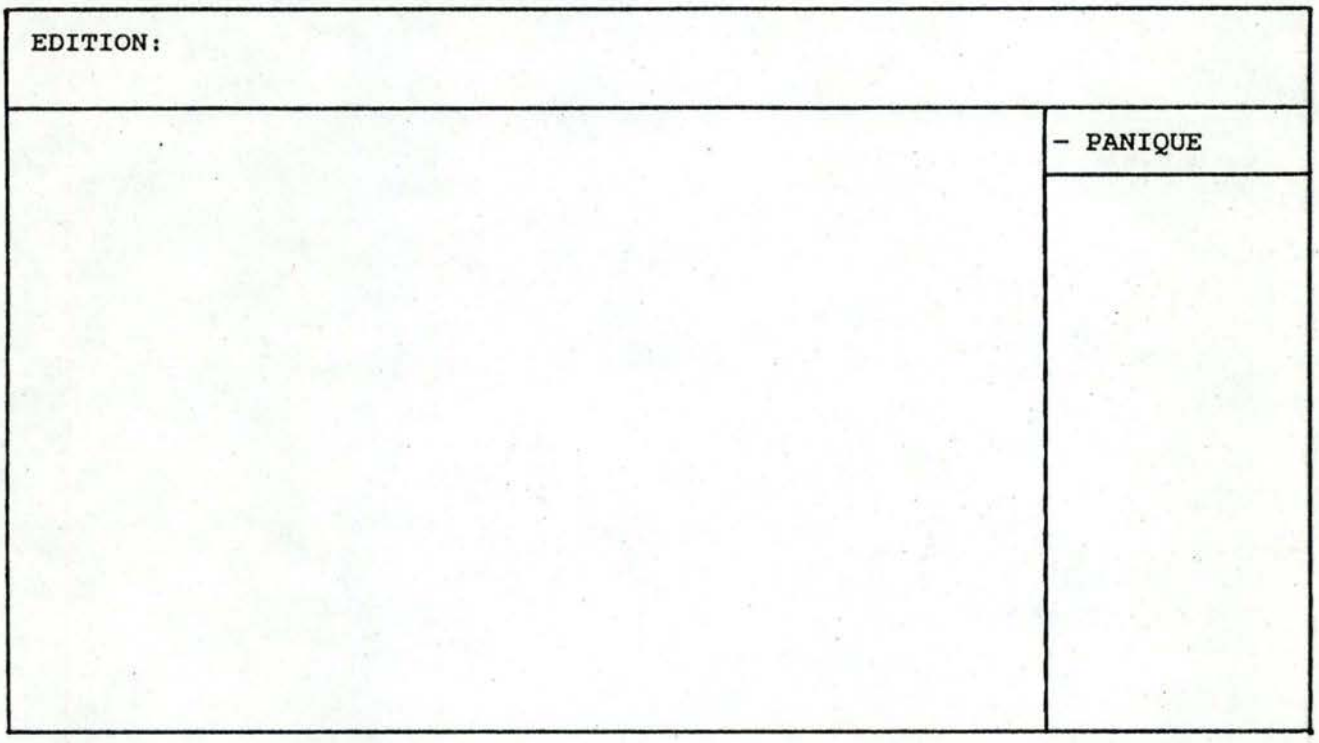
Ensuite, il décide des coordonnées de la fenêtre dans laquelle il désire qu'apparaisse le document choisi (voir paragraphe 2.3.2).

Le début du document apparaît alors dans la fenêtre et seuls des mouvements du curseur dans le texte sont possibles (cfr paragraphe 2.2).

2.9. L'action "panique"

L'option PANIQUE est un moyen rapide de sortir de l'éditeur qui permet de retrouver plus tard les mêmes documents dans l'état ou ils étaient lors de cette sortie.

Un nouvel écran apparaît où le précédent menu a été affacé et a été remplacé par l'écran ci-dessous :



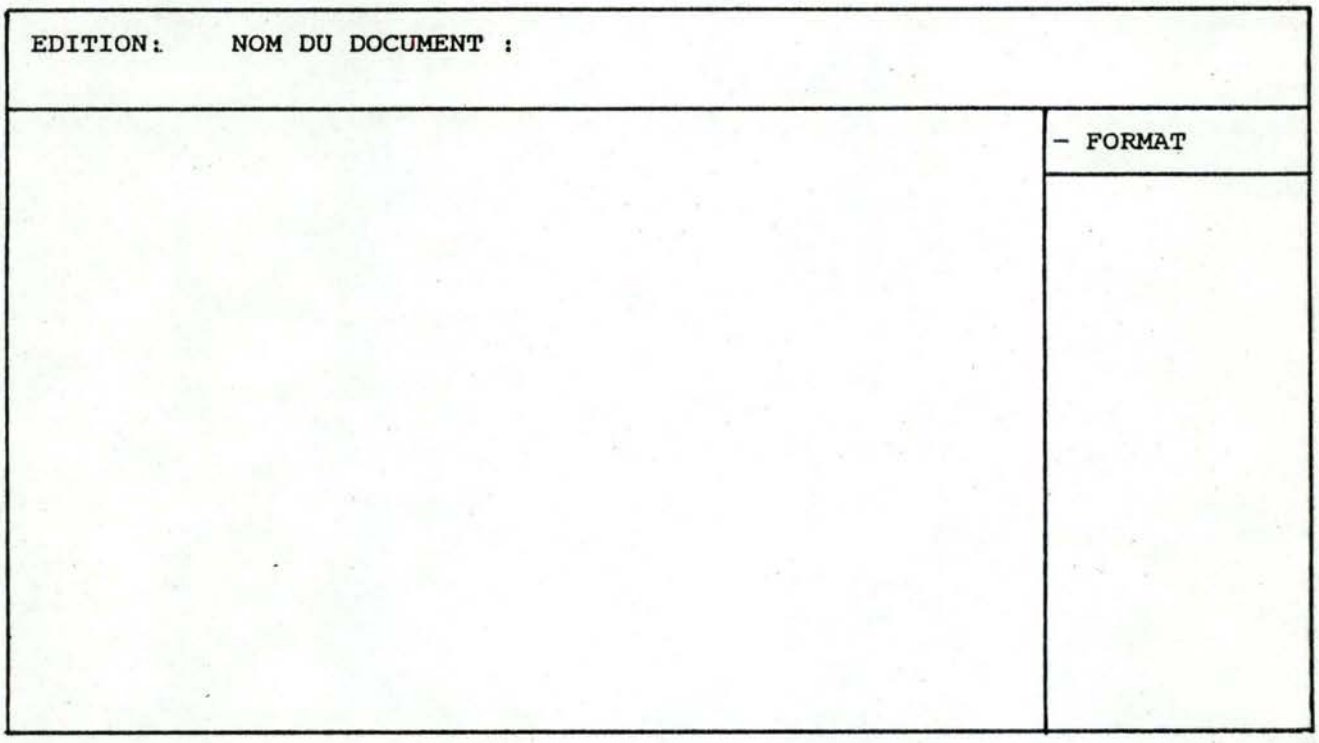
Le sauvetage de tous les documents modifiés dans une session est assuré.

Ensuite, il apparaît un dernier écran qui est l'écran de sortie (cfr paragraphe 5).

2.10. L'action "format"

L'option FORMAT affiche une liste de caractéristiques de formatage et d'impression du document dont l'identification est indiquée par l'utilisateur. Ce format existe lors de la création du document avec des valeurs par défaut qui peuvent être modifiées dans cette option.

Un nouvel écran apparaît où le précédent menu a été effacé et a été remplacé par l'écran ci-dessous :



Si l'utilisateur est entré par erreur dans la fonction, il lui suffit ,pour en sortir, de taper directement sur la touche <RETURN>.

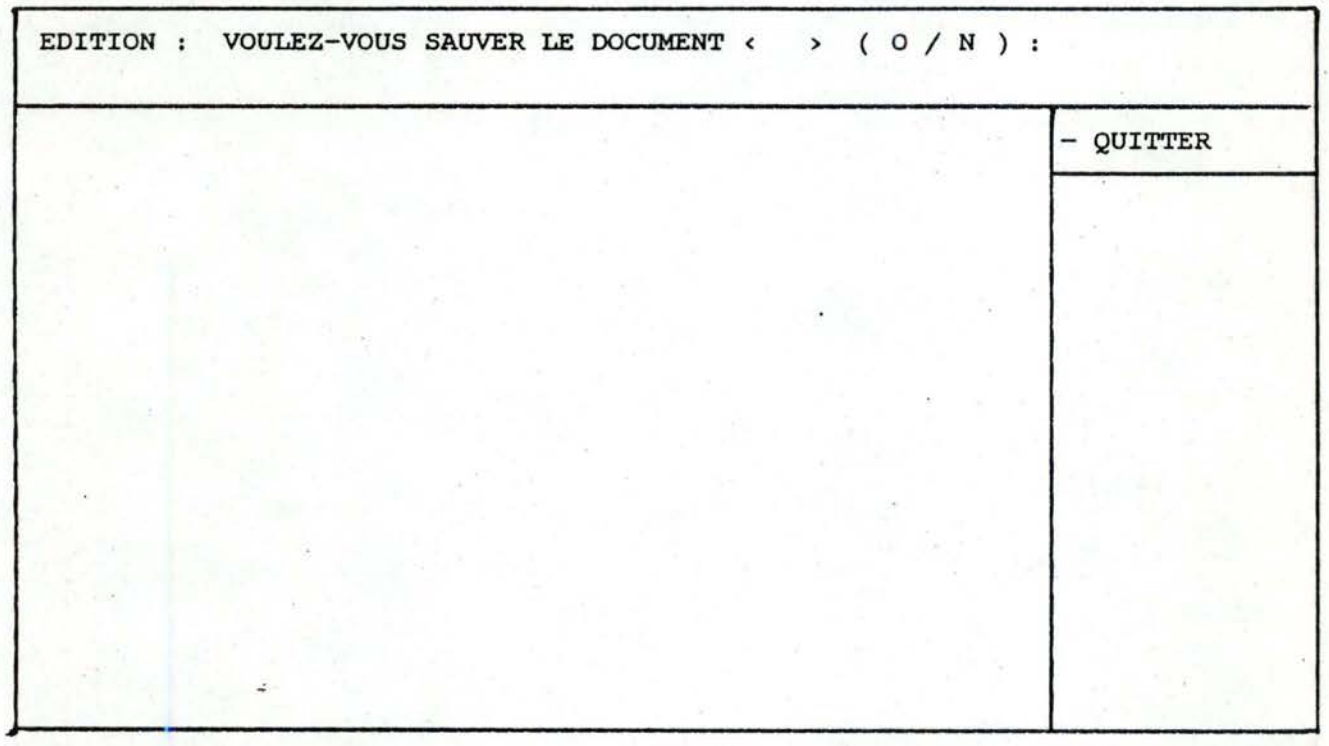
L'utilisateur introduit le nom du document dont il désire avoir les caractéristiques de formatage (voir paragraphe 2.3.1). Le curseur est placé automatiquement à droite du libellé "NOM DU DOCUMENT". Les renseignements apparaîtront selon la forme vue au paragraphe 2.3.3.

L'utilisateur peut maintenant, s'il le désire, modifier les caractéristiques de formatage du document (voir paragraphe 2.3.3).

2.11. L'action "quitter"

L'option QUITTER indique la fin de la phase d'édition de l'éditeur et permet de retourner au niveau supérieur. Dans cette fonction, il y a sauvetage physique des documents modifiés dont l'ordre de sauvetage a été donné et il y a destruction physique de ceux dont l'ordre de destruction a été donné. Les documents pour lequels aucun ordre n'a été donné ne seront pas sauvés. Toutefois, ceci ne se fera qu'après une demande de confirmation par le message "VOULEZ-VOUS SAUVER LE DOCUMENT < > (O / N): ".

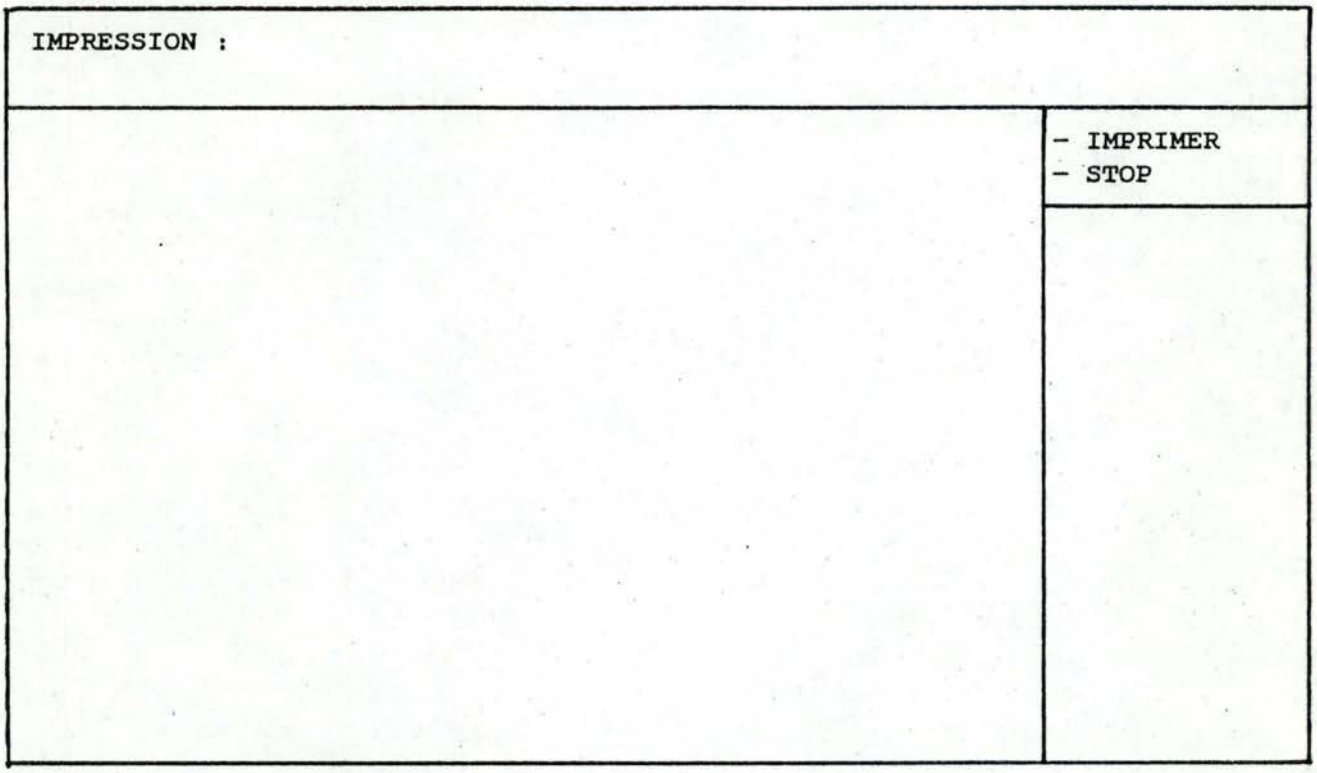
Ce n'est que dans ce dernier cas qu'un nouvel écran va apparaître où le précédent menu a été effacé et remplacé par le libellé "QUITTER".



Ensuite, l'utilisateur voit à nouveau apparaître le premier menu (cfr paragraphe 1).

3. L'impression (non implémenté)

Lors du choix de l'option IMPRESSION, l'écran suivant apparaît:



Le curseur est placé au début du menu et l'utilisateur dispose des touches de déplacement vertical du curseur pour déplacer le curseur dans le menu.

Ces touches sont :

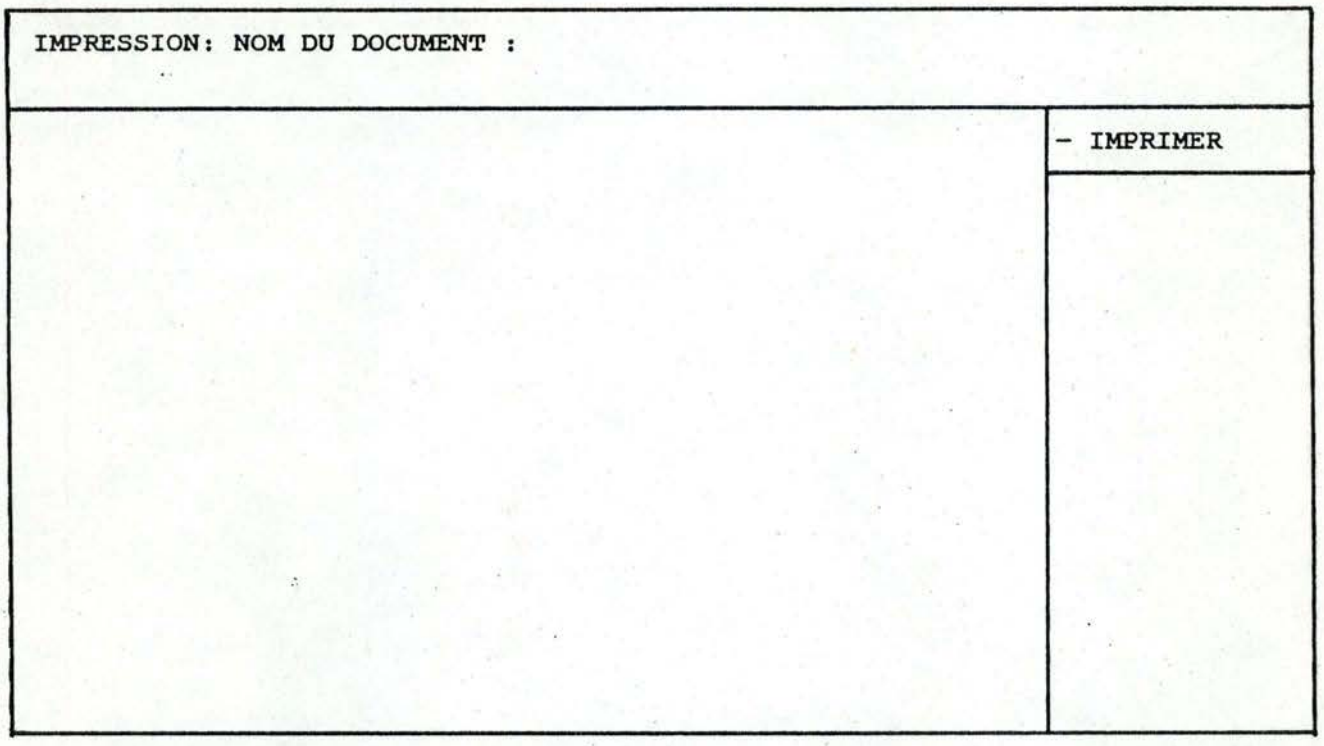
- CTRL / x
- CTRL / c
- CTRL / v
- CTRL / e
- CTRL / r
- CTRL / t

Une fois arrivé devant l'action choisie, l'utilisateur la sélectionne en poussant sur la touche ‹RETURN›.

3.1. L'action "imprimer"

L'option IMPRIMER permet d'imprimer sur une imprimante le document sous la forme décrite dans le format.

Un nouvel écran apparaît où le précédent menu a été effacé et a été remplacé par l'écran ci-dessous :



Si l'utilisateur est entré par erreur dans la fonction, il lui suffit ,pour en sortir, de taper directement sur la touche <RETURN>.

Ensuite, l'utilisateur indique le nom du document qu'il veut imprimer. (voir paragraphe 2.3.1) Le curseur est placé automatiquement à droite du libellé "NOM DU DOCUMENT".

3.2. L'action "stop"

L'option STOP termine la phase d'impression et permet de revenir au niveau supérieur.

L'utilisateur se retrouve dans le premier menu, c'est-à-dire l'écran suivant :

```
EDITEUR

                        M E N U
                        =======

                    _ E(dition)

                    _ I(mpression)

                    _ R(etour-panique)

                    _ S(top)


                        VOTRE CHOIX ? : _
```

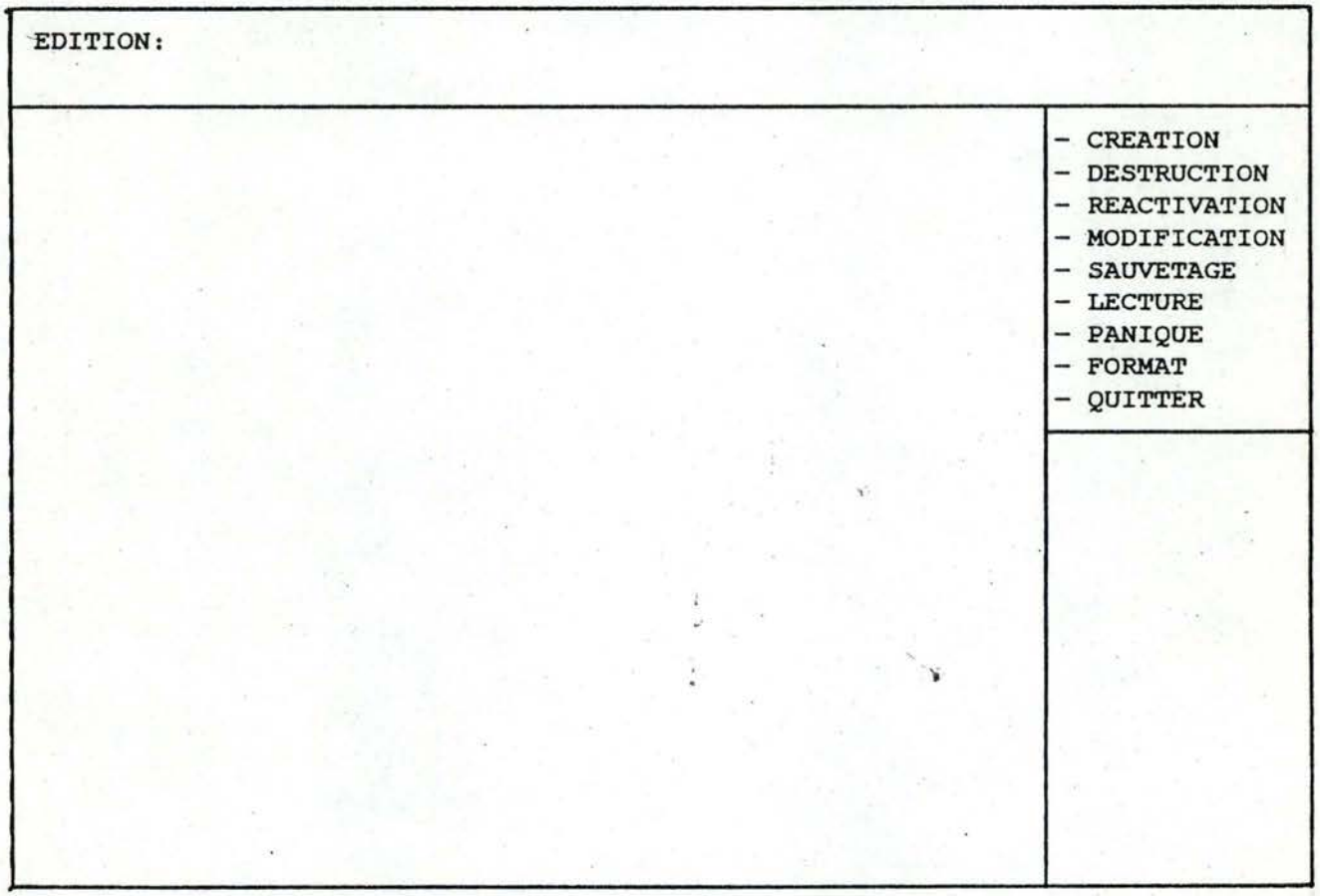
4. Le retour-panique

Une liste des documents manipulés lors de la précédente session apparaît sur l'écran. Ces documents sont caractérisés par leur état, c'est-à-dire s'ils sont détruits (logiquement), sauvés (logiquement), actifs (le dernier document sur lequel l'utilisateur a travaillé) ou sans rien de spécial.

Cette option permet de retrouver une situation quittée par le choix PANIQUE dans l'édition. Ce choix permet de retrouver tous les documents dans l'état où ils étaient lors du choix PANIQUE lors d'une session précédente.

En cas d'erreur, c'est-à-dire si l'utilisateur n'a pas quitté l'édition dans une session précédente par le choix PANIQUE, le message " VOUS N'ETES PAS SORTI EN PANIQUE LORS D'UNE SESSION PRECEDENTE DE L'EDITEUR" apparaît.

Lorsque la liste des documents manipulés est apparue, l'utilisateur doit pousser sur la touche ‹RETURN› et il se retrouve ainsi directement au niveau de l'édition. L'écran suivant apparaît :



5. Le stop

Un dernier écran apparaît :

L'utilisateur sort à cet instant-là de l'éditeur.

Chapitre 6

PORTABILITE DU SYSTEME

Chapitre 6: Portabilité du système

(Françoise Fourny et Jean-Marc Cheu)

1. Introduction.

Dans le chapitre 6, nous allons parler du matériel sur lequel doit tourner le système de traitement de texte et expliquer quelles sont les modifications qu'il faut apporter au système implémenté afin qu'il puisse être implémenté sur le smaky 8.

Ce chapitre 6, est divisé en 2 parties. La première de ces deux parties est consacrée à la définition des deux matériels utilisés. Le premier des deux matériels est un microprocesseur 16 bits (le smaky 8) qui est produit par l'école polytechnique fédérale de Lausanne en collaboration avec Epsitec SA. Le second est le PDP 11/45, um mini-ordinateur qui est fabriqué par Digital Equipment.

La deuxième partie évoque les problèmes relatifs à la portabilité du système de traitement de textes d'un matériel sur l'autre. Et elle explique quelles seront les parties du système de traitement de textes qui seront touchées par les changements.

2. Le matériel.

2.1. Smaky 8.

2.1.1. Description générale du Smaky 8.

Le smaky 8 est le dernier né d'une série de micro-ordinateurs développés par le LAMI de l'EPFL : son prédécesseur est un micro équipé du processeur Z 80 seulement (SM6). Le smaky 8 est un développement conjoint de l'EPFL (Prof. Nicoud) et d'Epsitec System SA. Il équipera le LAMI de l'EPFL et est commercialisé à partir de cette année.

Le smaky 8 est un micro-processeur comportant une unité centrale avec deux processeurs dont un seul peut travailler à la fois : un processeur 16 bits M68000 (4 ou 8 Mhz) et un Z 80 A (4 Mhz). Le processeur Z 80 A n'est qu'un outil de mise au point du hardware du Smaky 8. Ce processeur est destiné à exécuter les programmes existants sur le SM6 afin de tester la nouvelle configuration. Le Z 80 A est destiné à disparaître à long terme.

Le smaky 8 comporte une mémoire vive de 128 ou 256 KB. L'écran du smaky 8 est un écran demi-page interlacé, avec graphique et alphanumérique superposables. L'écran graphique a 512 lignes de 768 points et l'écran alphanumérique 32 ou 16 lignes de 96 caractères, définis chacun par un mot de 16 bits. Un point de l'écran correspond à un bit de la mémoire bit map.

La mémoire bit map a une capacité de 128 KB, et peut donc mémoriser deux pages graphiques. Une mémoire "bit map" est une mémoire qui grâce à un dispositif particulier permet de définir que cette mémoire reflète l'état de l'affichage à l'écran. Il suffit ainsi pour écrire sur l'écran de charger ce qu'il faut afficher dans la mémoire "bit map".

Le clavier du smaky 8 est un clavier suisse Romand(QWERTZ) avec clavier numérique, touches fonction et touches programmables.

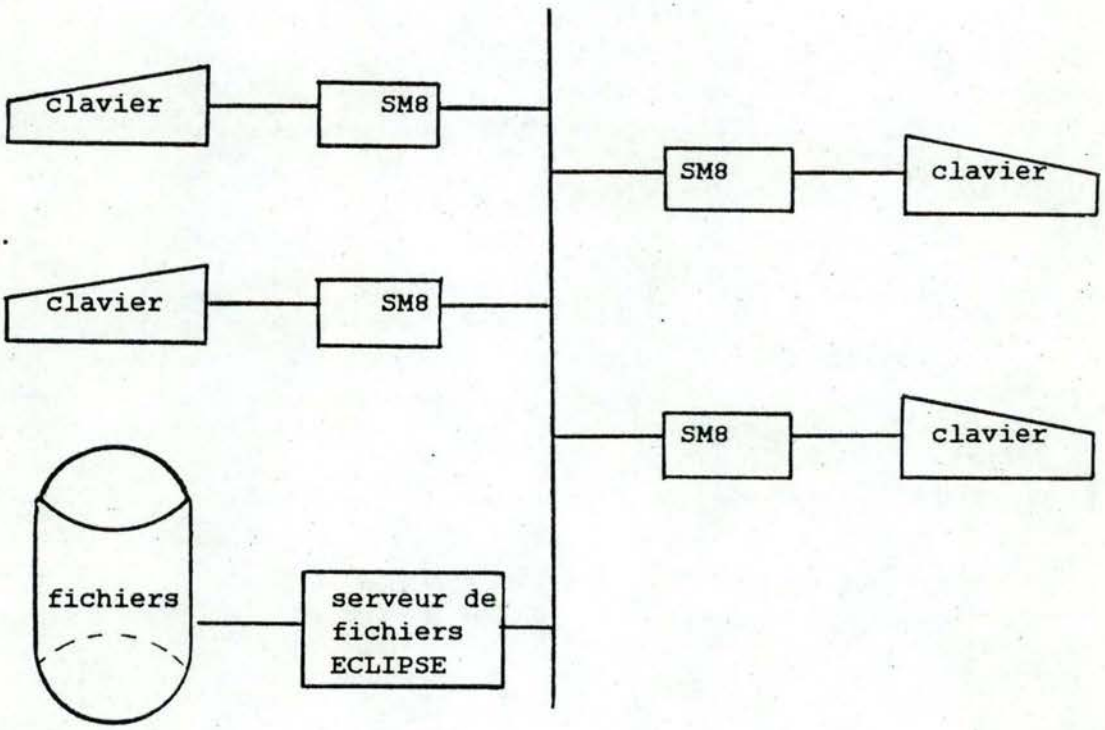
Le smaky 8 a un interface pour une souris ou un deuxième clavier.

La plupart des périphériques du commerce peuvent être connectés au smaky 8, en particulier tous ceux ayant un interface série ou parallèle. La réalisation d'interfaces spéciaux est aisée, et plusieurs lignes d'interruption sont réservées pour ces interfaces supplémentaires.

Un système d'exploitation de type UNIX est recherchée pour le smaky 8. Des compilateurs PASCAL, C, Modula, PORTAL, BASIC, FORTH, COBOL, FORTRAN seront implémentés dès que possible, avec des facilités de développement adéquat. La première génération de logiciel

sera dèrivée du logiciel actuel du smaky 6.

2.1.2. Le réseau.



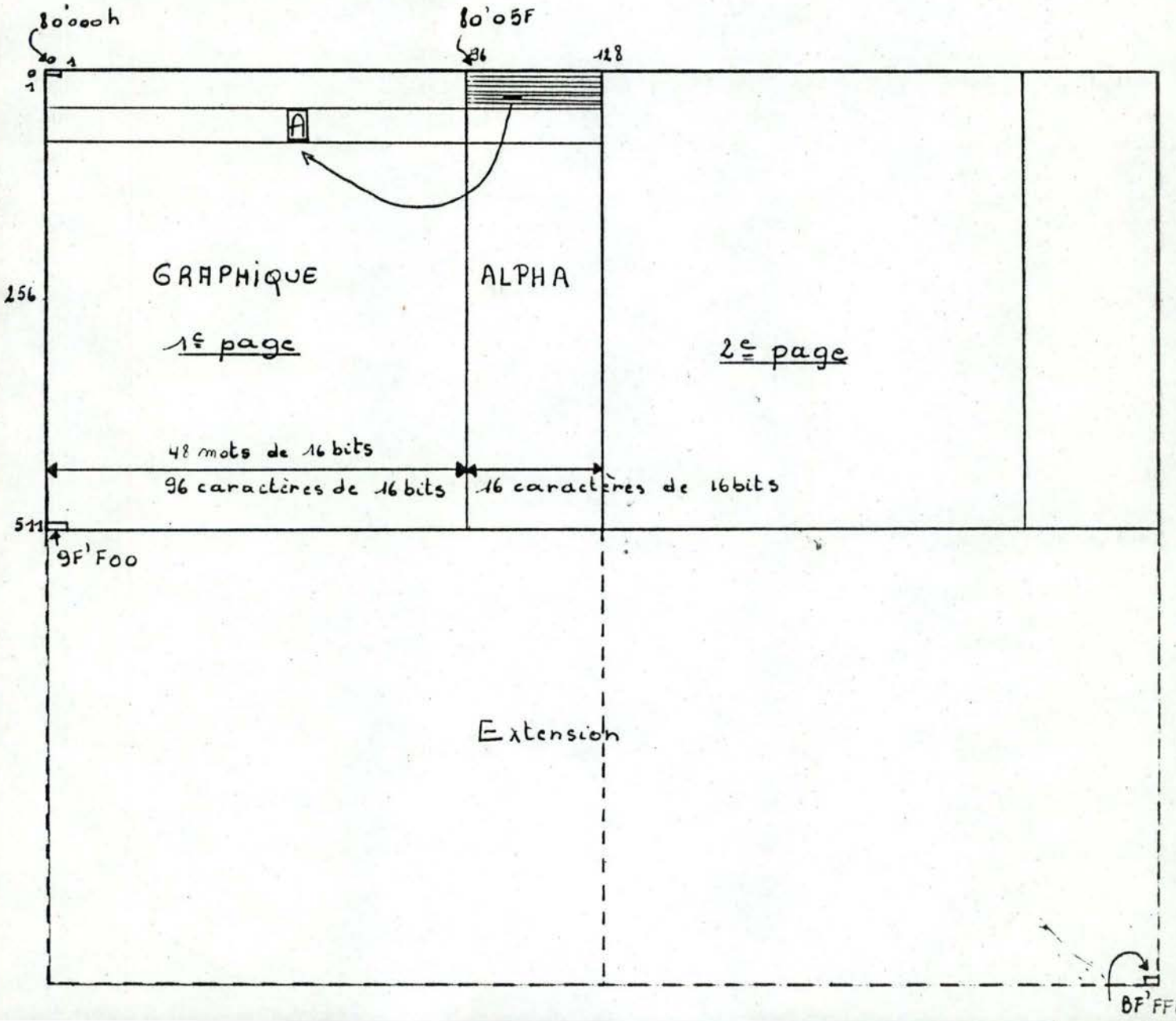
2.1.3. L'écran.

L'écran du smaky 8 montre le contenu d'une mémoire de 32 K mots de 16 bits divisée en une zone de "bit map" et une zone de "character map"; les 2 zones graphique et alphanumérique peuvent être montrées individuellemmt ou superposées.

Dans l'écran du smaky 8, il y a en fait 2 mémoires d'écrans côte à côte du point de vue des adresses (mais consécutives du point de vue du balayage). Le programmeur peut donc préparer une page graphique pendant que la précédente est affichée, ou faire du défilement sur 2 pages.

Les adresses graphiques sont telles que les 8 bits de poids faible correspondent à l'abscisse (valeurs significatives 0 à 95) et les 9 bits de poids fort à l'ordonnée (valeurs 0 à 511). Le point (0,0) correspond au coin supérieur gauche de l'écran (adresse 80000 hexa). La 2ème page correspond au bit 7 actif (80080). L'écran affiche 512 lignes en entrelacé.

Schéma des adresses de l'écran.

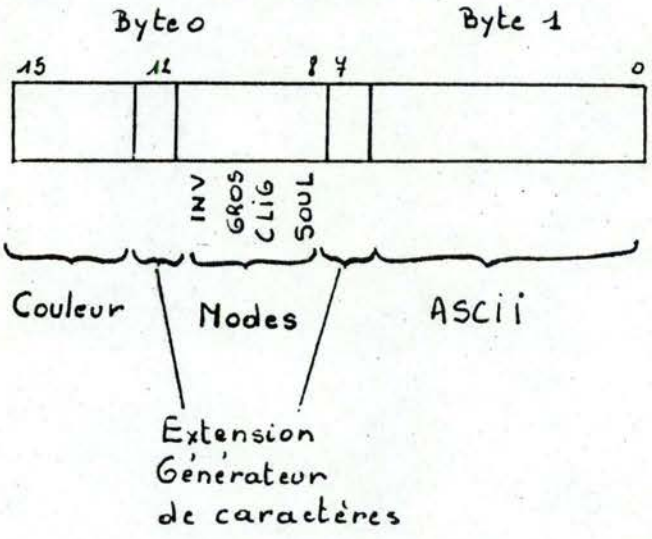


Les caractères alphanumériques sont définis par 16 bits et ont une résolution de 5 X 7 ou 7 X 9 dans une matrice de 8 X 16 points. Ceci conduit à un maximum de 32 lignes de textes de 96 caractères sur l'écran.

Les 16 bits associés à chaque caractère affiché sont répartis comme suit :

- code ASCII 7 bits
- police de caractère 2 bits
- mode d'affichage 4 bits
- réserve pour couleur 3 bits

Le format des codes des caractères alphanumériques est donné à la figure suivante.



2.2. Le PDP 11/45 et son environnement

2.2.1. Le PDP 11/45

Le PDP 11/45 est un mini-ordinateur de 256 KB de mémoire centrale. Dans sa configuration à l'Institut d'Informatique des Facultés Notre-Dame de la Paix, il est couplé à un processeur spécial pour les calculs en virgule flottante (FP-11). Il dispose en guise de mémoire auxiliaire :

- d'un disque dur de 64 MB (PM DS11/80)
- de trois disques de 2,5 MB (RK-05)
- de deux dérouleurs de bandes magnétiques d'une densité de 800 bpi (bit par inch), l'un ayant 9 pistes et l'autre 7.

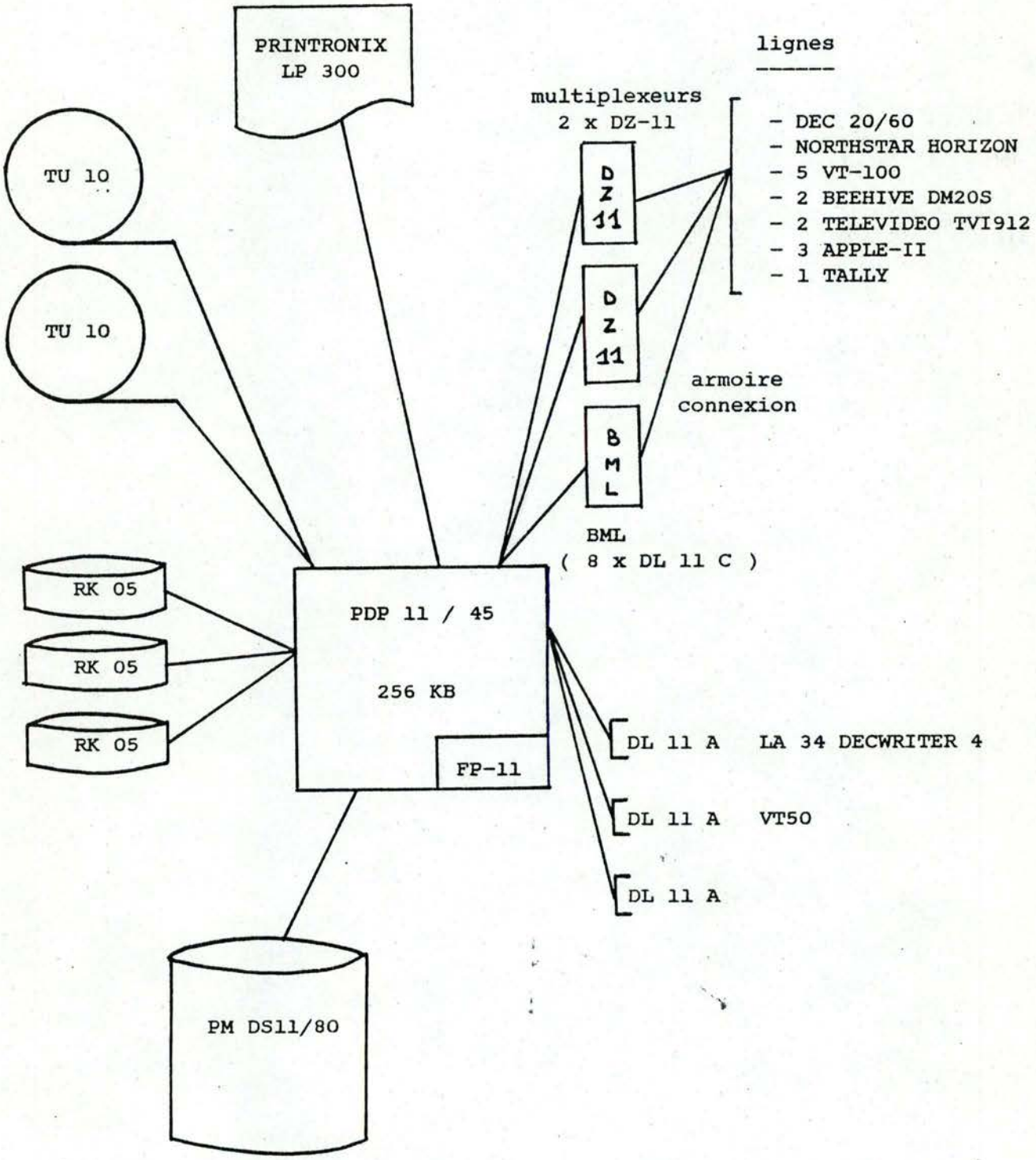
Le PDP possède une imprimante d'une vitesse d'impression de 300 lignes par minute (PRINTRONIX LP300).

Il dispose, via deux multiplexeurs (DZ-11) de 8 portes asynchrones chacun et via 11 interfaces asynchrones supplémentaires, au travers d'une armoire de connexion, de 27 lignes de connexions de terminaux. Une de ces lignes est reliée directement à un autre ordinateur, le DEC 2060, et deux autres à un micro-ordinateur, de type NORTHSTAR HORIZON II. A une bonne partie des autres lignes sont raccordés en étoile des terminaux vidéos aussi différents que des DEC VT-100, des TELEVIDEO 912, des APPLE-II ou des BEEHIVE DM20S. Un télétype TALLY est aussi raccordé à ce réseau.

Via deux des interfaces asynchrones (DL 11), un vidéo (VT-50) et un télétype servent de consoles en salle machine.

En résumé, nous pouvons faire un schéma de tous les appareils rattachés d'une manière ou d'une autre au PDP 11/45 (cfr le schéma de réseau).

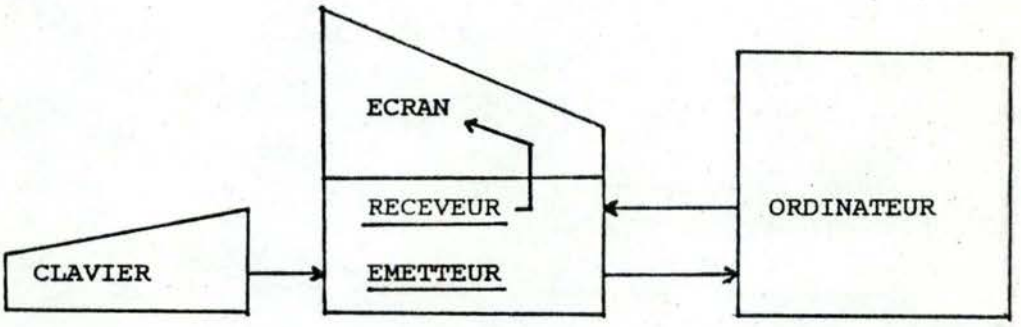
2.2.2. Le schéma du réseau



Nous allons maintenant entrer un peu plus dans les détails en décrivant le terminal VT-100 qui, dans l'état actuel de l'éditeur, est le seul sur lequel il est implémenté. Nous utilisons la version de base du VT-100, la version VT-100 AB.

2.2.3. Le VT-100

Comme tout terminal vidéo d'ordinateur, le VT-100 a une double fonction. Il est un outil pour envoyer à l'ordinateur via un émetteur de l'information introduite par le clavier. Simultanément, il est un moyen pour afficher à l'écran les réponses venant de l'ordinateur via un receveur (cfr figure ci-dessous).



Nous allons passer en revue les deux composants les plus visibles du VT-100, c'est-à-dire le clavier et l'écran.

2.2.3.1. Le clavier

Le clavier contient 83 touches (fonctions, alphabétiques, numériques). Ces touches sont divisées en un clavier principal de 65 touches dont l'arrangement est similaire à celui du clavier de machine à écrire standard et un clavier auxiliaire de 18 touches. Ces dernières sont :

- des touches numériques
- des touches de fonctions
- le point, la virgule et le signe moins.

Le clavier comporte aussi 7 indicateurs visuels:

- les 3 premiers indiquent si le terminal est en liaison avec l'ordinateur (ON-LINE) ou pas (LOCAL) ou si le clavier n'envoie pas d'information à l'ordinateur (KBD LOCKED).
- les 4 derniers sont programmables par l'utilisateur.

2.2.3.2. L'écran

L'écran mesure 12 pouces en diagonale et peut, au choix, afficher 24 lignes de 80 caractères ou 12 lignes de 132 caractères. Ceux-ci sont représentés à l'écran par une matrice de 7 x 9 points. L'écran peut être soit noir avec des caractères blancs soit blanc avec des caractères noirs. L'ensemble des caractères affichables à l'écran contient les 96 caractères ASCII.

L'écran du VT-100 est un écran alphanumérique avec des possibilités graphiques. C'est pourquoi le VT-100 dispose aussi d'un jeu de caractères graphiques (cfr tab-1). Ces possibilités sont d'ailleurs minimes par rapport à celles d'un terminal graphique.

code octal	car	car. graphique	code octal	car	car. graphique
137	_	blanc	157	o	ligne horizon. 1
140	\	diamant	160	p	ligne horizon. 2
141	a	indic. erreur	161	q	ligne horizon. 3
142	b	Tab. Horizon.	162	r	ligne horizon. 4
143	c	Form Feed (FF)	163	s	ligne horizon. 5
144	d	Carriage Return	164	t	'T' gauche
145	e	Line Feed (LF)	165	u	'T' droit
146	f	symbole degre	166	v	'T' renversé
147	g	plus ou moins	167	w	'T' normal
150	h	Newline (NL)	170	x	barre verticale
151	i	Tab. Vertical	171	y	plus petit ou egal
152	j	coin inf. droit	172	z	plus grand ou egal
153	k	coin sup. droit	173	{	Pi
154	l	coin sup. gau.	174	|	différent de
155	m	coin inf. gau.	175	}	la livre Sterling
156	n	croisem. lignes	176	~	point centré

tab-1

NB : il y a cinq lignes horizontales qui diffèrent par la hauteur à laquelle elles se trouvent.

Le curseur, quant à lui, peut au choix être un bloc clignotant ou un soulignement clignotant.

2.2.3.3. Les fonctions utilisées

Parmi une multitude de fonctions offertes par le VT-100, nous n'en avons utilisé que trois afin de permettre au programme d'être portable. Les trois fonctions utilisées sont vraiment indispensables. Ce sont le positionnement du curseur, l'effacement de l'écran et l'utilisation des possibilités graphiques du VT-100. Celui-ci offre à l'utilisateur deux modes de fonctionnement : le mode ANSI (VT-100) et le mode VT-52. Nous avons choisi le mode ANSI car celui-ci correspond à la norme ANSI X3.64-1977. En effet, beaucoup de terminaux sont compatibles au VT-100 par cette norme, ce qui permet une plus grande portabilité de l'éditeur. Il est évident que les trois fonctions choisies sont dépendantes du mode dans lequel elles sont utilisées.

Le positionnement du curseur est une fonction qui nous permet de déplacer le curseur à n'importe quel endroit de l'écran. Cette fonction, très utile pour la gestion de l'écran, n'est jamais prévue dans un langage. On fait toujours appel à des procédures en librairie, utilisant d'une manière ou d'une autre le type du terminal.

L'effacement de l'écran est une fonction qui nous permet quasi-instantanément d'effacer tout l'écran. Ceci donne à l'éditeur une plus grande vitesse pour imprimer l'écran suivant. C'est donc beaucoup plus agréable pour l'utilisateur.

L'utilisation des possibilités graphiques du VT-100 est, dans le cas de l'éditeur que nous avons conçu, la fonction la plus spectaculaire. Les écrans imprimés sont beaucoup plus beaux grâce au jeu de caractères graphiques dont dispose le VT-100. Nous utilisons notamment les caractères qui nous permettent de dessiner des cadres.

3. Portabilite.

Nous étudions ci-dessous les modifications à apporter au système de traitement de textes que nous avons réalisé afin que celui-ci soit exécutable sur le Smaky 8 alors qu'il a été implémenté sur le PDP-11/45. Ces modifications sont peu nombreuses car dans la structure des modules, nous avons essayé de tenir compte des contraintes des deux matériels. Nous avons donc essayé de faciliter et minimiser, dès le début, les modifications à apporter. Pour ce faire, nous avons essayé de localiser les endroits sujets à des modifications dans des modules séparés et même dans des fichiers sources séparés du reste des programmes. Nous avons aussi essayé de n'utiliser du langage C que ce qui est le plus standard sans utiliser des facilités permises par la version utilisée. Nous étions à même de faire cela car nous avons pu travailler avec deux versions du langage C (la version 6 sur le PDP-11 à NAMUR et la version 7 sur le VAX à LAUSANNE).

Tout d'abord, une premières exigences à ce transport, est le fait qu'il doit y avoir sur le Smaky 8 le langage C. La version du compilateur doit être semblable à celle que nous avons utilisée sur le PDP-11. Si, le Smaky 8 ne supporte pas le C, il faudra en premier lieu adapter le compilateur C sur le Smaky 8. Si, d'autre part, le C supporté par le Smaky 8 est différent du C avec lequel nous avons travaillé, il faudra modifier le programme afin de le convertir en la version du C supportée par le Smaky 8.

Une autre solution plus simple à ce problème est d'utiliser un cross-compilateur C sur le Smaky8. Un cross-compilateur est un compilateur qui génère à la compilation un langage dont le format des instructions est directement compréhensible par une machine. En l'occurrence, le cross-compilateur générerait un code 68000 exécutable sur le processeur 68000 du Smaky8.

En dehors de problèmes du langage, d'autres modifications sont à apporter aux programmes. Un premier type de modifications est celui concernant l'affichage à l'écran. En effet, on a pu voir dans la partie de ce chapitre parlant du matériel que l'écran du VT-100 est fort différent de l'écran du Smaky 8 : le terminal du PDP-11 est un VT-100 et l'écran du Smaky 8 est une mémoire bit map. Sur l'écran du Smaky 8, pour afficher quelque chose à l'écran, il suffit de charger à la bonne adresse ce que l'on veut afficher. Sur le VT-100 il faut faire une entree-sortie sur le périphérique écran. Il résulte donc de cette différence qu'il faudra modifier toute la gestion de l'écran.

Un deuxième type de modifications à apporter au système de traitement de textes, sont les initialisations des constantes utilisées dans la gestion d'écran. Ces constantes servent à définir les bornes de l'écran pour le positionnement du curseur, le dessin de la fenêtre, les affichages. De plus, certaines définitions sont à détruire complètement car elles ne présentent plus aucun intérêt pour l'implémentation des programmes sur le Smaky 8.

Un troisième type de modifications concerne les modes du terminal. Ces modes sont : RAW,ECHO,TAB,GRAPHIC. Tous peuvent être combinés.

- RAW est un mode permettant le passage immédiat au programme des données lues au clavier, c'est-à-dire que l'on n'attend pas un vidage du buffer et chaque caractère lu est transmis directement au programme. Le mode inverse (NO-RAW) est le mode où il faut attendre un vidage du buffer pour pouvoir traiter les caractères lus au clavier. Sur le Smaky 8, il faudrait réaliser aussi cette fonction du terminal.

- ECHO est un mode où l'écho sur l'écran de chaque caractère entré au clavier est fait par l'ordinateur. Ce mode peut être inversé, et à ce moment, les caractères entrés au clavier n'apparaissent pas sur l'écran. Dans le cas du travail sur le Smaky 8, puisque l'affichage à l'écran se fait simplement en chargeant la position mémoire adéquate dans la mémoire écran, il suffit d'utiliser une zone de la mémoire différente de la mémoire écran quand l'on ne désire pas l'écho et d'utiliser une zone de la mémoire écran quand l'on désire l'écho.

- TAB est un mode du terminal où quand l'utilisateur entre un tab, le caractère ascii tab est envoyé. NO_TAB est le mode inverse du terminal. Dans ce mode, quand un caractère tab est envoyé, le gestionnaire du terminal envoie un nombre de blancs. Ce nombre de blancs est égal au nombre de caractères qu'il faut pour arriver à la position du tabulateur suivant.

- GRAPHIC est le mode graphique du VT-100. Dans ce mode on peut écrire certains caractères spéciaux qui donne des caractères graphiques simples à l'affichage sur l'écran (voir tableau dans paragraphe sur le VT-100). Sur le Smaky 8, il a aussi des possibilités de graphisme. Elles sont plus grandes que sur le VT-100 car on dispose de deux pages écrans : une page graphique, une page alphanumérique. ASCII, le mode alphanumérique du VT-100, est l'inverse de GRAPHIC. Dans ce mode, on ne peut afficher que des caractères alphanumériques classiques (les caractères d'imprimerie classiques). Le Smaky 8 permet aussi de passer en mode alphanumérique pur. Dans les deux systèmes, l'affichage d'une page de l'écran permet le mélange des modes graphique et alphanumérique. Dans le cas du VT-100, il suffit de passer en graphique pour écrire ce qui est en graphique et de passer en alphanumérique pour écrire des caractères. Sur une même ligne, il peut y avoir un mélange des deux modes. Dans le cas du Smaky8, il suffit de travailler en mode entrelacé. Ce mode entrelacé permet de travailler à la fois en graphique et en alphanumérique.

On peut donc remarquer que les seules modifications à apporter au système, à part les problèmes de langage, sont des modifications concernant l'écran car ceci est la seule différence des deux configurations vue par le programme.

CONCLUSION

Conclusion.

Ce mémoire avait pour objectif, l'étude des systèmes de traitement de textes présents sur le marché et l'analyse et l'implémentation d'un éditeur particulier dont les caractéristiques ont été définies lors de notre stage à l'Ecole Polytechnique fédérale de Lausanne.

Nous avons, dans le premier chapitre de ce mémoire, étudié un grand nombre d'éditeurs existant sur le marché. Chacun de ces éditeurs a vu ses caractéristiques principales mises en exergue. Un tableau récapitulatif a permis d'avoir une vue générale sur tous ces éditeurs.

Dans les deux chapitres suivants, nous avons expliqué la démarche d'analyse du système à réaliser.

Dans le premier de ces deux chapitres, les caractéristiques de l'éditeur à réaliser ont été définies et nous avons fait l'analyse fonctionnelle du système. Rappelons que l'analyse d'un projet informatique compte 3 grandes étapes :

- -L'analyse conceptuelle (FAUT-IL LE FAIRE).
- -L'analyse fonctionnelle (QUE FAIRE).
- -L'analyse organique (COMMENT FAIRE).

Nous n'avons pas fait d'analyse conceptuelle : nous en avons donné les raisons dans l'introduction du chapitre concernant l'analyse fonctionnelle.

Le second de ces deux chapitres a été consacré à l'analyse organique du système de traitement de textes. L'analyse organique complète du système a été laissée en annexe, mais les choix suivants ont été expliqués dans ce chapitre : le langage utilisé pour programmer, l'architecture du système, le support physique utilisé pour mémoriser un fichier et l'organisation du fichier sur ce support dans l'implémentation.

Le quatrième chapitre a parlé des méthodes d'implémentation et du planning des tests. Ces méthodes d'implémentation recouvrent deux aspects : la méthode de programmation et la méthode de test. La méthode utilisée était la méthode descendante avec à un certain moment une utilisation de la méthode ascendante. Dans la partie planning des tests, nous avons parlé de la répartition des tests dans l'implémentation des modules.

Le cinquième chapitre constitue le manuel d'utilisateur du système implémenté. La façon d'utiliser l'outil réalisé y est expliquée en détail.

Le dernier chapitre évoque le problème de portabilité posé dans l'implémentation du sytème. En effet, le système développé sur un PDP-11 doit être adapté Sur un Smaky 8. Dans ce chapitre , les 2 matériels différents ont été présenté. Et dans une partie de ce chapitre nous avons parlé des modifications à apporter au système pour l'adapter sur le Smaky 8.

Pour terminer ce mémoire, nous évaluons le travail effectué quant à la réalisation du système de traitement de textes. Nous ne prétendons pas avoir réalisé le meilleur éditeur, mais nous pensons que la façon dont nous avons pensé celui-ci permet à l'utilisateur une plus grande facilité d'apprentissage et d'utilisation. Rappelons les caractéristiques principales que nous avons voulu donner à l'éditeur réalisé :

- Un interface facile et simple pour un utilisateur. Cet interface est réalisé principalement par l'apparition permanente de menus sur l'écran.
- Un éditeur "fidèle" c'est-à-dire ayant une représentation à du document à l'écran la plus proche possible de ce qui sera imprimé.

Par souci de présenter un minimum de programmes implémentés, nous avons du simplifier l'éditeur et c'est pourquoi nous proposons toute un série d'améliorations que l'on pourra apporter ultérieurement à cet éditeur.

- Prévoir la gestion d'écran adaptée à différent types de terminaux (VT-100, VT-52, VISUAL 200, ...).

- Améliorer la primitive DEFAIRE en constituant une pile des X dernières opérations effectuées sur le texte. Cela afin de pouvoir revenir au maximum X étapes en arrière à partir d'un état (le dernier) du document. On pourrait aussi prévoir de déterminer un état (n'importe lequel) comme étant celui de base et, quand on revient en arrière, on ne pourrait plus déplacer cet état. A l'heure actuelle, cette primitive n'effectue que l'aiguillage d'un document vers une ancienne version de ce document.

- Etoffer la phase Retour-panique. La procédure a été simplifiée à l'extrême. Elle ne fait que l'affichage de la liste des documents édités dans la session précédente alors qu'il serait intéressant de repasser directement dans le mode Edition et de continuer, s'il y a lieu, la dernière opération avant le choix Panique.

- Dans le module de formatage, les nouvelles données entrées par l'utilisateur doivent être cadrées à droite dans la zone d'entrée de la donnée. Cette contrainte pourrait être supprimée pour plus de facilité de l'utilisateur.

- Prévoir le traitement des symboles mathématiques.

- Implémenter les phases fonctions et primitives qui ne seront pas implémentées.

Ces modules non implémentés sont :

- le module d'impression
- le module de lecture
- le module montre
- le module copie
- le module caractère
- le module encadre
- le module pagine
- le module visualise
- le module minuscule
- le module majuscule
- le module efface
- le module transfert

- Prévoir l'interface souris pour le déplacement du curseur.

- Faire apparaître les messages dans une fenêtre et non plus toujours à la même place, ce qui est le cas maintenant.

- Avoir un menu qui apparait sous le curseur sur commande et non plus un menu fixe qui occupe pas mal de place sur l'écran.

- Augmenter la rapidité de traitement et surtout d'affichage de l'éditeur.

- Faire l'écho nous-mêmes afin de ne pas détruire l'écran s'affichant en tapant des commandes à l'avance.

- Masquer les interruptions afin de ne pas permettre de sortie de l'éditeur autre que par la sortie normale.

- Incorporer les touches "flèche" pour le déplacement ou utiliser d'autres touches que celles utilisées afin de ne pas perturber les habitudes déjà prises par les utilisateurs.

- Mettre les fichiers temporaires dans la directory TMP pour qu'ils soient gérés automatiquement.

- Prévoir une procédure de visualisation d'une page du document dans la fonction de lecture afin d'avoir une vue globale de chaque page du document.

- Permettre le choix dans un menu par une lettre au lieu du déplacement du curseur.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[1] Meyrowitz N. et Van Dam A. " Interactive editing system: A survey "
August 5 1981

[2] Kernighan B. W. " A tutorial introduction to the UNIX text editor "
Bell Laboratories September 1978

[3] Kernighan B. W. et Ritchie D. M. " The C programming language "
Prentice Hall 1978

[4] The Seybold Report " Xerox'star "
vol. 10 no 16 April 27 1981

[5] Bobst Graphic " Manuel d'utilisation du BG-1000 (BEEZY) "

[6] Bertrand P. " Thot, un éditeur-formateur de texte "
université de Liège (Faculté des sciences appliquées) Mars 1980

[7] Robson D. " Object oriented software system "
Learning Research Group (PARC) Byte Publication August 1981 p 74 -> 86

[8] Tesler L. " The Smalltalk Environment "
Apple Computer Inc Byte Publication August 1981 p 90 -> 147

[9] Laboratoire de microinformatique (EPFL) " Smaky 8 "
Epsitec system SA Aout 1981

[10] Ebel N. "Cours modulaire de programmation : annexe 3"
(Manuel d'utilisation de l'éditeur EDIS)
EPFL (département mathématique) 1981

[11] Wegmann A. " Introduction au PASCAL UCSD "
Epsitec system SA MARS 1980 p 17 -> 25

[12] Alto User's Handbook : " BRAVO manual "
octobre 1976 p 27 -> 58

[13] Massachussets Institute of Technology "EMACS manual for TOPS-20 users"
EMACS version 142 6 february 1980

[14] Digital Equipment Corporation " User guide for VT-100 "
january 1979

[15] Hainaut JL "Fichiers et Banque de données"
Un modèle descriptif de bases de données au niveau organique : le modèle d'accès JANVIER 81

[16] Irons E.T. et Djorup F.M. "A CRT Editing System"
Communication ACM , 1 (january 1972) p 16 -> 20

[17] Sobemap La bureautique Le traitement de textes
DOC 700/1 Bruxelles

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

RUE GRANDGAGNAGE, 21, B - 5000 NAMUR (BELGIUM)

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR,

INSTITUT D'INFORMATIQUE

SYSTEME DE TRAITEMENT

DE TEXTES.

(annexes)

Promoteur : Ph. Van Bastelaer.

Françoise Fourny.
Jean-Marc Cheu.

Mémoire présenté en vue
de l'obtention du grade de

LICENCIE ET MAITRE EN INFORMATIQUE

ANNEE ACADEMIQUE 1981 - 1982.

ANNEXE A

ANALYSE ORGANIQUE

Annexe A: Analyse organique.

(Fourny Françoise)

1. Introduction.

L'analyse organique succède à l'analyse fonctionnelle. Dans l'analyse fonctionnelle, nous avions défini QUE faire maintenant dans l'analyse organique nous définissons COMMENT FAIRE. La description de l'analyse organique comporte 3 volets : une description des modules, une description des messages intervenant et une description des données utilisées.

Nous définissons dans l'analyse organique pour chaque module ses entrées, ses sorties, sa description, le traitement qu'il effectue, le pseudo-code associé à ce traitement et la dynamique qui anime ce traitement.

Les entrées comprennent, pour chaque module, les messages et les données dont il a besoin pour effectuer son traitement.

Les sorties comprennent les messages et les données produites par le traitement.

Dans la description du module, nous donnons l'objectif global de celui-ci en quelques lignes.

Dans la partie traitement du module, les principales règles de traitement sont données sous la forme de table de décision ou sous la forme de phrases françaises.

Dans la partie relative à la dynamique du module, nous expliquons les mécanismes de déclenchement du module et donnons la liste des modules qu'il déclenche.

Dans la partie concernant les prê- et postconditions, nous exprimons les conditions qui doivent être vérifiées à l'entrée et la sortie du module. Toute les prê- et postconditions ne sont pas reprises pour chaque niveau. Nous avons, pour chaque niveau, repris les prê- et postconditions du niveau précédent et celles du niveau auquel appartient le module analysé.

Dans la partie pseudo-code, la logique du module est exprimée en un pseudo-langage non exécutable. Ce pseudo-code sera la base utilisée pour le codage de chaque module en C. La totalité des modules a été transformée en pseudo-code , même les modules non-implémentés. Il est possible de continuer a implémenter les modules non-implémentés en se basant sur l'analyse organique, et plus particulièrement sur le pseudo-code de chacun des modules.

Pour chaque message, nous donnons son nom et une description soit schématique soit phraséologique.

Dans la partie concernant la description des données, nous donnons une définition précise de celles-ci.

2. Description des modules.

Application : editeur-appl.

Synonyme.

Système de préparation de documents

Sous-partie.

Initialisation.

Entrée.

Edit-command

Sortie.

Un ensemble de documents

Module : Initialisation.

Entrée.

Choix-util-0

Sortie.

Menu-initialisation.

Sous_parties.

Impression-ph-1,
Edition-ph-2,
Retour-panique-fct.

Dynamique.

Cette phase est déclenchée par le message Edit-commande. Cette phase reçoit le message choix-util-0 et génère le message Menu-initialisation. Elle déclenche ensuite selon le contenu du message reçu soit la phase Impression-ph-1, la phase Edition-ph-2, la phase Retour-panique-fct.

Description.

Cette module détermine vers quelle autre module de l'éditeur on va se diriger.

Précondition.

Il existe une commande d'édition.

Postcondition.

(choix-util-0 appartient {e,E,i,I,r,R,s,S}) AND (Il existe une commande d'édition)

Traitement.

choix-util-0	I	E	R	S	différent
déclencher impression-ph-1	X				
déclencher édition-ph-2		X			
déclencher ret-pan-fct			X		
arrêt de l'éditeur				X	
erreur					X

Pseudo-code.

```
begin
  receive "edit-commande";
  afficher ecran;
  get choix;
  while choix "not equal" STOP OR S DO
       begin
                if choix = I OR i
                   begin
                        CALL IMPRESSION;
                        goto endb;
                   end
                if choix = E OR e
                   begin
                        CALL EDITION;
                        goto endb;
                   end
                if choix = R OR r
                   begin
                        CALL RETOUR-PANIQUE;
                        go to endb;
                   end
                CALL ERREUR (1)
       endb     afficher ecran;
                get choix;
 .end
```

Module : Impression-ph-1

Entrée.

Choix-util-1

Sortie.

Document-imprimé,
Ecran-impr,
Document imprimé.

Sous_parties.

Impression-fct-1.

Dynamique.

Cette phase est déclenchée à la fin de la phase Initialisation si le contenu du message Choix-util-0 est I ou Impression. Elle génère un message : Ecran-impr. et produit un document imprimé. Elle reçoit le message choix-util-1 et déclenche selon le contenu du message Choix-util-1 la fonction Impression-fct-1 ou permet de revenir au niveau supérieur.

Description.

Cette phase permet d'imprimer un ou plusieurs documents suivant le format qui lui est associé.

Précondition.

(choix_util_0= I ou i) AND (il existe une edit_command)

Postcondition.

(choix_util_0= I ou i) AND (il existe une edit_command) AND ((choix_util_1 = Impression) OR (choix_util_1 = Stop))

Traitement.

choix_util_1	I	S	#
déclencher impression	X		
revenir au niveau supé.		X	
erreur			X

Pseudo-code.

```
begin
        affich_impress( );
        get choix;
        while choix # retour do
                begin
                   imprimer(nom_doc);
                end
end
```

Module : impression-fct-1.

Entrée.

Nom-document,
Document.

Sortie.

Document imprimé.

Dynamique.

Cette fonction est déclenchée si le contenu du message choix- util-1 est I ou Impression. Elle reçoit le message Nom-document et produit le document imprimé.

Description.

Cette fonction imprime le document choisi sur une imprimante s'il existe. Le document sera imprimé selon les caractéristiques contenues dans le enregistrement "format".

- la marge détermine l'espace blanc à gauche à partir du bord de la feuille par défaut 25mm
- la largeur de texte détermine le nombre maximum de caractères que l'on va imprimer sur une ligne par défaut 80 caractères
- l'interligne détermine la largeur de l'interligne
  1 = l'interligne normal
  2 = interligne double (on écrit une ligne de blanc)
  3 = interligne triple (on écrit deux lignes de blancs)
  4 = interligne quadruple (on écrit 3 lignes de blancs)
- la marge haut détermine l'espace blanc en haut à partir du bord de la feuille par défaut 20mm
- la fin de page détermine le nombre de lignes contenues sur une page par défaut 60 lignes
- les tabulateurs désignent les positions à laquelle se placera successivement le curseur en poussant la touche TAB
- la fonte détermine le type de caractère normal que prendra le texte :
  -messenger (par défaut)
  -grec
  -helvesan
  -italique

Précondition.

(choix-util-0 = I ou i) AND (choix-util-1 = IMPRESSION)

Postcondition.

(choix-util-0 = I ou i) AND (choix-util-1 = IMPRESSION) AND ((document appartient à {document}) AND (document imprimé)) AND ((document n'appartient pas a{document}) AND (erreur))

Traitement.

Il faut tout d'abord acquérir le nom du document à imprimer et vérifier la validité de ce nom. Suivant le choix de l'utilisateur prendre les caractères dans la fonte adéquate.
Tant qu'on n'est pas à la fin du document :

```
        Si la ligne de texte lue signifie saut de page
                        - inscrire le numéro de page
                        - initialiser le compteur de lignes à
                          zero
                        - effectuer le saut de page
                        - incrémenter le numéro de page
        Sinon
                Si le nombre de ligne imprimée < MAXLINE
                        - incrémenter le compteur de lignes
                        - préparer la ligne de sortie
                        - écrire la ligne
                Si le nombre de lignes imprimées = MAXLINE
                        _ initialiser le nombre de lignes par
                          page à 1
                        - écrire le numéro de page sur
                          la feuille précédente
                        - sauter à la page
                        - incrémenter le nombre de lignes
                        - écrire la ligne
        Sauter (interligne - 1) lignes
```

Pseudo-code.

```
begin
        nblig=0;
        nbpage=1;
        if fonte = Messenger then prendre le générateur Messenger;
        if fonte = Helvesan then prendre le générateur Helvesan;
        if fonte = Italique then prendre le générateur Italique;
        if fonte = Grec then prendre le générateur Grec;
debut :affich_impr( );
        get (nom_doc);
        while nom_doc > 14
           begin
                erreur (3);
                get (nom_doc);
           end
        while nom_doc n'appartient pas à {document}
                begin
                   erreur(2);
                   get(nom_doc);
                end
        read document into ligne-e;
        while not EOF do
        begin
           move all spaces into lignes-s;
           for (i=0;i<longmax;++i) ligne-s(i+marge_gauche)= ligne(i);
           add 1 to nblig;
           write ligne-s after interligne;
           if (nblig>bas de page) OR (dernier caract= FF)
```

```
                        begin
                           if pagine then
                                begin
                                   write ligne-page after 2;
                                   nbpage=nbpage+1;
                                end
                           new-page;
                           nblig=0;
                     read document into ligne-e;
                  end
                  close document;
            end
```

Module: Edition-ph-2

Entrée.

choix-util-2

Sortie.

Documents-édités,
DESCRTAB.

Sous-parties.

Creation-fct-2,
Destruction-fct-2,
Activation-fct-2,
Modification-fct-2,
Sauvetage-fct-2,
Panique-fct-2,
Lecture-fct-2,
Format-fct-2,
Quitter-fct-2.

Dynamique.

Cette phase est déclenchée à la fin de la phase Initialisation si le contenu du message Choix-util-0 est E ou Edition. Elle génère un message : Ecran-édit et recoit le message choix-util-2. Elle déclenche selon le contenu du message Choix-util-2 les fonctions suivantes : Creation-fct-2,
Destruction-fct-2,
Activation-fct-2,
Modification-fct-2,
Sauvetage-fct-2,
Panique-fct-2,
Lecture-fct-2,
Format-fct-2,
Quitter-fct-2.

Description.

Objectifs : Cette phase permet l'édition proprement dite d'un document.

Précondition.

(choix-util-0 = E OR e) AND (il existe une edit-commande)

Postcondition.

(choix-util-1 appartient à {C,D,A,M,S,F,P,R,L}) AND (choix-util-0 = E OR e) AND (il existe une edit-commande)

Traitement.

choix-util-1	C	D	A	M	S	F	P	R	L	#
Déclencher création	X									
Déclencher destruction		X								
Déclencher activation			X							
Déclencher modification				X						
Déclencher sauvetage					X					
Déclencher format						X				
Déclencher panique							X			
Déclencher retour								X		
Déclencher lecture									X	
Erreur										X

Pseudo-code.

```
begin
  afficher écran;
  lect_choix( );
  while choix "not equal" retour do
        begin
            if choix = C then call création ;
                              go to endb;
            if choix = D then call destruction;
                              go to endb;
            if choix = M then call modification;
                              go to endb;
            if choix = A then call activation;
                              go to endb;
            if choix = P then call panique;
                              go to endb;
            if choix = S then call sauvetage;
                              go to endb;
            if choix = F then call format;
                              go to endb;
            if choix = L then call lecture;
                              go to endb;
            erreur(1);
      endb : afficher écran;
            get choix;
        end
end
```

Module : Création-fct-2

Entrée.

Nom-document,
Choix-util-2-c.

Sortie.

Document,
DESCRTAB,
Ecran-creat.

Sous-parties.

Justifie-fct-2,
Pagine-fct-2,
Vis-car-spec-fct-2,
saut-de-page-fct-2,
Montre-fct-2,
Copie-fct-2,
Cherche-fct-2,
Change-fct-2,
Minuscule-fct-2,
Majuscule-fct-2,
Transfert-fct-2,
Caractère-fct-2,
Defaire-fct-2,
Encadre-fct-2,
Efface-fct-2,
Centre-fct,
Depl-1-haut-fct-2;
Depl-1-bas-fct-2,
Depl-4-haut-fct-2,
Depl-4-bas-fct-2,
Depl-deb-txt-fct-2,
Depl-fin-txt-fct-2,
Depl-car-gau-fct-2,
Depl-car-dr-fct-2,
Depl-mot-gau-fct-2,
Depl-mot-dr-fct-2,
insertion-fct-2,
del-fct-2,
BS-fct-2,
formatage,
gestion-prim.

Dynamique.

Cette fonction est déclenchée si le contenu du message Choix- util-2 est C ou Création. Elle génère deux messages : Ecran- creat et Document-creat.Elle reçoit le message Choix-util-2-c. Elle déclenche ensuite selon le contenu du message Choix-util-2-c les fonctions suivantes : Justifie-fct-2,
Pagine-fct-2,
Vis-car-spec-fct-2,

Saut-de-page-fct-2,
Montre-fct-2,
Copie-fct-2,
Cherche-fct-2,
Change-fct-2,
Minuscule-fct-2,
Majuscule-fct-2,
Transfert-fct-2,
Caractère-fct-2,
Defaire-fct-2,
Encadre-fct-2,
Efface-fct-2,
Centre-fct,
Retour-fct-2,
Depl-1-haut-fct-2,
Depl-1-bas-fct-2,
Depl-4-haut-fct-2,
Depl-4-bas-fct-2,
Depl-deb-txt-fct-2,
Depl-fin-txt-fct-2,
Depl-car-gau-fct-2,
Depl-car-dr-fct-2,
Depl-mot-gau-fct-2,
Depl-mot-dr-fct-2,
Insertion-fct-2,
Del-fct-2,
BS-fct-2.

Description.

Cette fonction permet la création proprement dite d'un document. Règles:
- acquisition du nom de document et vérification de sa validité
- acquisition des coordonnées de la fenêtre
- dessin de la fenêtre
- création de l'entête et des caractéristiques de formatage
  initialiser les attributs de l'entête du document
    - nom du document
    - la date de création (dd mmm hh:mm)
    - la date de mise à jour (dd mmm hh:mm)
    - code du document selon son type
  modififier si nécessaire les caractéristiques de formatage.
- début de la création proprement dite du texte (gestion des primitives).

Précondition.

(choix-util-0 = E OR e) AND (choix-util-1 = Création)

Postcondition.

((document à créer appartient à { documents } (erreur)) AND (document à créer n'appartient pas à { document } (créer document)) AND (choix-util-0 = E ou e) AND (choix-util-2 = Création)

Traitement.

nom-doc appartient à {documents}	Y	Y	Y	Y	N	N	N	N
long_doc > longueur max.(14)	Y	Y	N	N	Y	Y	N	N
nom-doc = vide	Y	N	Y	N	Y	N	Y	N
Impossible	X	X	X		X			
Erreur : existe déjà				X				
Erreur : trop long						X		
Créer fenêtre								X
Vérif. poss. de formatage								X
Créer FMT								
Primitives								X
Créer fichier								X
Sortir création							X	X

Pseudo-code.

```
begin
                receive DESCRTAB from edition-ph-2;
    début :     afficher création_écran;
                setcurs entête écran;
                get (nom document);
                if nom document = "  " go to fin-création;
                if longueur-nom-document > 14
                        then erreur ('3')
                             effacer le nom du document;
                             go to début;
                if nom-document appartient à {document}
                        then erreur ('4')
                             effacer le nom du document;
                             go to début;
                créate (nom-document);
                get fichier (nom-document);
                création {FORMAT (enreg.) du nom-document;
                         {ENTETE
                vérification et modification des valeurs FORMAT;
                gestion des primitives;
end
```

Module: Modification-fct-2

Entrée.

Nom-document,
document.
choix-util-2-m

Sortie.

écran-mod.
Document modifié,
Descripteur du document(DESCRTAB).

Sous-parties.

Justifie-fct-2,
Pagine-fct-2,
Vis-car-spec-fct-2,
Saut-de-page-fct-2,
Montre-fct-2,
Copie-fct-2,
Cherche-fct-2,
Change-fct-2,
Minuscule-fct-2,
Majuscule-fct-2,
Transfert-fct-2,
Caractère-fct-2,
Defaire-fct-2,
Encadre-fct-2,
Efface-fct-2,
Centre-fct,
Depl-1-haut-fct-2,
Depl-1-bas-fct-2,
Depl-4-haut-fct-2,
Depl-4-bas-fct-2,
Depl-deb-txt-fct-2,
Depl-fin-txt-fct-2,
Depl-car-gau-fct-2,
Depl-car-dr-fct-2,
Depl-mot-gau-fct-2,
Depl-mot-dr-fct-2,
Insertion-fct-2,
Del-fct-2,
BS-fct-2,
gestion-prim.

Dynamique.

Cette fonction est déclenchée si le contenu du message Choix- util-2 est M ou Modification. Elle génère deux messages : Ecran- mod et Document-mod. Elle reçoit le message Choix-util-2-m. Elle déclenche ensuite selon le contenu du message Choix-msg-2-m les fonctions suivantes : Justifie-fct-2,
Pagine-fct-2,

Vis-car-spec-fct-2,
Saut-de-page-fct-2,
Montre-fct-2,
Copie-fct-2,
Cherche-fct-2,
Change-fct-2,
Minuscule-fct-2,
Majuscule-fct-2,
Transfert-fct-2,
Caractère-fct-2,
Defaire-fct-2,
Encadre-fct-2,
Efface-fct-2,
Centre-fct,
Retour-fct-2,
Depl-1-haut-fct-2,
Depl-1-bas-fct-2,
Depl-4-haut-fct-2,
Depl-4-bas-fct-2,
Depl-deb-txt-fct-2,
Depl-fin-txt-fct-2,
Depl-car-gau-fct-2,
Depl-car-dr-fct-2,
Depl-mot-gau-fct-2,
Depl-mot-dr-fct-2,
Insertion-fct-2,
BS-fct-2,
Del-fct-2.

Description.

Objectifs :
Cette fonction permet la modification proprement dite d'un document.

Règles :
- acquisition et vérification du nom de document
- mise à jour de la date de mise à jour
- Si le document a déjà été édité lors de cette session, restaurer les renseignements de la fenêtre
  Sinon acquérir ces renseignements et puis mettre le document dans la zone de travail
- Modification du texte (gestion des primitives)

Précondition.

(choix-util-0 = E ou e) AND (choix-util-2 = Modification)

Postcondition.

(choix-util-0 = E ou e) AND (choix-util-2 = Modification) AND (si document à modifier appartient {document} alors modifier document) AND (si document à modifier n'appartient pas à {document} alors ERREUR)

Traitement.

nom doc app. à {doc}	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
long-doc > long-max	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
nom doc vide	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
déjà édité	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
impossible	X	X	X		X		X		X	X	X		X			
n'existe pas								X								
nom-doc trop long				X								X				
créer fenêtre																X
mettre doc en workfile																X
mise à jour entête															X	X
primitives															X	X
sortir						X								X	X	X

Pseudo-code.

```
begin
                receive (DESCRTAB) from edition-ph-2;
    début : afficher écran-mod;
                setcurs entête écran;
                get (nom document);
                if nom document = "  " go to fin-création;
                if longueur-nom-document > 14
                        then erreur ('3')
                             effacer le nom du document;
                             go to début;
                if nom-document n'appartient pas à {document}
                        then erreur ('2')
                             effacer le nom du document;
                             go to début;
                get fichier (nom-document);
                mise-à-jour ENTETE (enreg.) du nom-document;
                gestion des primitives;
end
```

Module: Destruction-fct-2

Entrée.

Nom-document

Sortie.

écran-dest
DESCRTAB
flag D
flag S

Dynamique.

Cette fonction est déclenchée si le contenu du message Choix- util-2 est D ou Destruction. Elle génère le message Ecran-dest.

Description.

Objectifs :
Cette fonction permet de détruire logiquement un fichier. La destruction réelle n'interviendra qu'à la fin de la session d'édition.
Règles :
- acquisition et vérification du nom du document
- mise à 0 du flag S et mise à 1 du flag D de la table des descripteurs si le nom du document se trouve dans celle-ci sinon erreur.

Précondition.

(choix-util-0 = E ou e) AND (choix-util-3 = D ou c).

Postcondition.

(choix-util-0 = E ou e) AND (choix-util-3 = D ou c) AND (descripteur-document-s = 0) AND ((descripteur-document-d = 1) AND (nom-document valable)) OR (erreur AND (nom-document non valable))

Traitement.

document appartient à {document}	Y	Y	Y	Y	N	N	N	N
nom-document <= 14 (long)	Y	Y	N	N	Y	Y	N	N
nom-document appartient à descripteurs	Y	N	Y	N	Y	N	Y	N
flag D = 1	X							
flag S = 0	X							
clear fenêtre	X							
erreur		X	X	X	X	X	X	X

Après avoir simplifié la table, on obtient une table réduite :

document appartient à {document}	Y	Y	Y	N
nom-document <= 14	Y	-	N	-
nom-document appartient à {descr.}	Y	N	Y	-
flag D = 1	X			
flag S = 0	X			
clear fenêtre	X			
erreur		X	X	X

Pseudo-code.

```
begin
        receive (DESCRTAB) from edition-ph-2;
        afficher écran-dest;
début : lire nom-fichier;
        si pas de nom de fichier go to fin-des;
        if nom-fich > 14 then
                        begin
                             erreur(3);
                             clear nom-doc;
                             go to déb;
                        end
        if nom-fich n'appartient pas à directory
                              begin
                                      erreur(2);
                                      clear nom-doc;
                                      go to déb;
                              end
        DESCRTAB.D correspondant =1;
        clear entête, écran;
fin-des : end;
```

Module: Activation-fct-2

Entrée.

Nom-document

Sortie.

écran-act
flag A
flag S
fenêtre

Dynamique.

Cette fonction est déclenchée si le contenu du message Choix-util-2 est A ou Activation. Elle génère le message Ecran-act.

Description.

Objectifs : Cette fonction permet d'annuler l'ordre de destruction d'un document et de rendre visible la fenêtre désignée.

Règles :

- acquisition et vérification du nom du document
- Si le document a déjà été édité lors de cette session
  restaurer les renseignements de la fenêtre
  sinon acquérir ces renseignements
  mettre le document en zone de travail.

Précondition.

(choix-util-0 = E OR e) AND (choix-util-3 = A)

Postcondition.

(choix-util-0 = E OR e) AND (choix-util-3 = A) ((descripteur-document.a = 1) AND (nom-document valable)) OR (erreur AND nom-document pas valable))

Traitement.

Active un document c-à-dire il y a un changement du document courant en cours de traitement.

document appartient à {document}	Y	Y	N	N
document appartient à DESCRTAB.	Y	N	Y	N
restaurer Rens.fen.	X	X		
faire fenêtre	X	X		
afficher doc. dans fenêtre	X	X		
flag A = 1	X	X		
mettre fichier dans un work-file		X		
erreur (impossible)			X	X

Pseudo-code.

```
begin
        affiche-activation;
        get (nom-document);
        while (nom-document n'appartient pas à {document})
           begin
                 erreur(2);
                 get (nom-document);
           end
        if nom-document n'appartient pas à DESCRTAB
                        then mettre nom-document en work-file;
        restaurer RENS_FEN;
        créer fenêtre;
        effacer l'intérieur de la fenêtre;
        afficher nomdocument dans la fenêtre;
        DESCRTAB.a correspondant = 1;
end;
```

Module : Lecture-fct-2.

Entrée.

Document

Sortie.

écran-lect
Fenêtre document
DESCRTAB

Dynamique.

Cette fonction est déclenchée si le contenu du message Choix-util-2 est L ou Lecture. Elle reçoit deux messages : Choix-msg-2-1 et génère le message fenêtre-lect. Ensuite, selon le contenu du message Choix-util-2-1, elle déclenche les fonctions suivantes : Depl-1-haut-fct-2,
Depl-1-bas-fct-2,
Depl-4-haut-fct-2,
Depl-4-bas-fct-2,
Depl-deb-txt-fct-2,
Depl-fin-txt-fct-2,
Depl-car-gau-fct-2,
Depl-car-dr-fct-2,
Depl-mot-gau-fct-2,
Depl-mot-dr-fct-2.

Description.

Objectifs :
Cette fonction effectue la lecture du document dont le nom aura été donné. Cette fonction va imprimer à l'écran le document choisi si ce document existe. L'utilisateur pourra se déplacer dans le texte grâce aux primitives de déplacement. Ces déplacement sont : _ déplacement avant du texte dans la fenêtre _ déplacement arrière du texte dans la fenêtre _ déplacement latéral gauche de la fenêtre _ déplacement latéral droit de la fenêtre.

Règles : Cette fonction effectue l'affichage d'un document dans une fenêtre dont les coordonnées auront été définies par l'utilisateur. Ce dernier peut déplacer un curseur dans la fenêtre de texte grâce aux routines de déplacement du curseur. Dans le cas où l'utilisateur entre un caractère différent de ceux des déplacements de curseur, il ne se passe rien. Pour sortir, il faut entrer RETURN.

Choix-utilisateur	E	R	T	S	D	F	G	X	C	V	#
depl-1-haut		X									
depl-1-bas									X		
depl-4-haut	X										
depl-4-bas								X			
depl-deb-txt			X								
depl-fin-txt										X	
depl-car-gau					X						
depl-car-droit						X					
depl-mot-gau				X							
depl-mot-droit							X				
rien faire											X

Précondition.

(choix_util_0=E OR e) AND (choix_util_2=L)

Postcondition.

(choix_util_0=E OR e) AND (choix_util_2=L) AND (document-lu appartient à {document})

Traitement.

document-lu appartient à {document}	Y	N
imprimer à l'écran	X	
erreur		X
clear écran (fenêtre)	X	
primitives de déplacement	X	

Pseudo-code.

```
begin
        receive DESCRTAB from édition-ph.;
        get (nom-doc.);
        while (nom-doc > 14)
          begin
                erreur ('3');
                get nom-doc;
          end
        while (k < 20)
          begin
             if nom-doc = DESCRTAB [k]  then {trouvé = TRUE;
                                              break;
                                             }
             ++ k;
```

```
        end
      if (trouvé = FALSE)
        then
          begin
            get fich( );
            DESCRTAB.s = 1;
            DESCRTAB.d = 0;
            DESCRTAB.a = 1;
          end
      depl-deb-txt (nom-doc);
      while (getcla != return) do
        begin
              if CTRL-X
               begin
                then
                      call depl-4-bas;
                      go to endw;
               end
              if CTRL-T
               begin
                then
                      call depl-fin-txt;
                      go to endw;
               end
              if CTRL-E
               begin
                then
                      call depl-4-haut;
                      go to endw;
               end
              if CTRL-U
               begin
                then
                      call deb-txt;
                      go to endw;
               end
              if CTRL-C
               begin
                then
                      call depl-1-bas;
                      go to endw;
               end
              if CTRL-R
               begin
                then
                      call depl-1-haut;
                      go to endw;
               end
         endb: end
end
```

Module: Sauvetage-fct-2

Entrée.

Nom-document

Sortie.

écran-sauv.
flag S,flag D

Dynamique.

Cette fonction est déclenchée si le contenu du message Choix-util-2 est S ou Sauvetage. Elle génère le message Ecran-sauv.

Description.

Objectifs :
Cette fonction permet de sauver les modifications apportées à un fichier.
Règles :
- acquisition et vérification de la validité du nom du document.
- Si le document n'a pas été édité lors de cette session, alors erreur
  sinon mettre le flag S à 1 et le flag D à 0.

Précondition.

(choix-util-0 = E OR e) AND (choix-util-3= D OR d)

Postcondition.

(choix-util-0 = E OR e) AND (choix-util-2= S OR s) AND ((descripteur-document-S = 1) AND (descripteur-document-d = 0) AND (nom-document valable) AND (nom-document appartient a {document})) OR (erreur)

Traitement.

document appartient à {document}	Y	N	Y	N
nom-document valable	Y	Y	N	N
sauver (met flag S = 1)	X			
erreur		X	X	X
clear fenêtre	X			

Pseudo-code.

```
begin
     receive DESCRTAB from edition-ph-2;
     affich_sauv( );
déb-sauv : lire nom-document;
             si pas de nom-document go to fin-sauv;
             si longueur nom-document > 14 then
                                        begin
                                           erreur (3);
                                           clear nom-document;
                                           go to déb-sauv;
                                        end
             si nom-document n'appartient pas à {document} then
                                        begin
                                           erreur (2);
                                           clear nom-document;
                                           go to déb-sauv;
                                        end
             si l'utilisateur désire sauver sous un nom
             différent du nom original
                                        begin
                                           lire le nouveau nom;
                                           recopier le fichier;
                                           sous le nouveau nom;
                                        end
            sinon  DESCRTAB.S correspondant =1  ;
            clear fenêtre document;
fin-sauv : end
```

Module: Panique-fct-2

Entrè.

DESCRTAB.

Sortie.

Ecran-pan,
Descripteur du document,
Documents en zone de travail.

Dynamique.

Cette fonction est dèclenchèe si le contenu du message Choix-util-2 est P ou Panique. Elle gènère le message Ecran-pan.

Description.

Objectifs :
Cette fonction permet de quitter rapidement l'èditeur sans que rien ne soit changè.

Règles :
Si aucun document en zone de travail,
sinon
- mèmoriser le nom et les caractèristiques nècessaires à la reprise de la session dans les mêmes conditions.
- garder tous les documents èditès dans la zone de travail.

Prècondition.

(choix-util-0 = E OR e) AND (choix-util-2 = P)

Postcondition.

(choix-util-0 = E OR e) AND (choix-util-2 = P) AND (il existe un fichier PANIQUE) AND (tous les fichiers temporaires sont sauvès)

Traitement.

Table descripteur vide	Y	N
recopier table des descr. dans fichier DESCR.PAN.		X
fermer les fichiers temporaires		X
erreur	X	

Pseudo-code.

```
begin
        receive DESCRTAB from edition-ph-2;
        affich-panique;
        initialisation : i = 0;
        créer le fichier PANIQUE;
        while (i < 20)
                begin
                   fermer le fichier TEMPORAIRE;
                   écrire le nom du fichier dans le fichier PANIQUE;
                end
        fermer le fichier PANIQUE;
end;
```

Module: Format-fct-2

Entrée.

Nom-document,
Document.

Sortie.

Ecran-fmt.
Format du document.

Dynamique.

Cette fonction est déclenchée si le contenu du message choix-util-2 est F ou Format. Elle génère le message Ecran-fmt.

Description.

Objectifs :
Cette fonction permet à partir de valeurs par défaut d'obtenir les caractéristiques de formatage d'un texte.

Précondition.

(choix-util = E ou e) AND (choix util = F ou f)

Postcondition.

(choix-util = E ou e) AND (choix util = F ou f) AND (nom_doc appartient à {documents}) AND (fonte appartient à {messenger, helvesan, grec, italique}) AND (Vérif., coupe, justifg, justifgd appartient à {OUI, NON}) AND (marge appartient à [0,100]) AND (bas de page appartient à [0,66]) AND (interligne appartient à [1,2,3,4]) AND (marge hauteur appartient à [0,66]) AND (tab appartient à [0,100]) AND (nombre de blanc entre deux mots appartient à [0,10])

Traitement.

Effectuer une vérification du nom de fichier entré lorsqu'il est correct il faut vérifier et modifier les valeurs de l'enreg. FORMAT.

Pseudo-code.

```
begin
        affiche-format;
        get (nom-document);
        while (nom-document n'appartient pas à {document}
                begin
                        erreur(2);
                        get (nom-document)
                end
         while (indtab <= max) OR (trouvé = TRUE)
                begin
                        if (nom-document = DESCRTAB.nom-document)
                                then trouvé = TRUE;
                        indtab = indtab + 1;
                end
        if trouvé=FALSE then
                begin
                  get-fich(recopier fichier sur temporaire);
                  DESCRTAB[ind] = nom-document;
                end
        formatage(nom-document);
end
```

Module: Quitter-fct-2

Entrée.

Descripteur de document,
Documents édités en zone de travail.

Sortie.

Ecran-quit
Documents.

Dynamique.

Cette fonction est déclenchée si le contenu du message Choix- util-2 est Q ou Quitter. Elle retourne ensuite au module Initialisation.

Description.

Objectifs :
Cette fonction permet de retourner au niveau supérieur, c-à-d à la phase Initialisation. A ce moment, on effectue le sauvetage des fichiers dont l'ordre de sauvetage a été donné et la destruction de ceux dont l'ordre de destruction a été donné.

Règles :
Pour chaque descripteur de document :
Si flag D = 1
- effacer le document de la zone de travail
- effacer le document édité

Si flag S = 1
- recopier la zone de travail sur le document édité
- effacer le document de la zone de travail

Si flag S = 0 et flag D = 0
- effacer le document de la zone de travail après avoir demandé à l'utilisateur si il faut le sauver ou pas.

Précondition.

(choix-util-0 = E OR e) AND (choix-util-2 = S OR s)

Postcondition.

(choix-util-0 = E OR e) AND (choix-util-3 = S OR s) AND (Pour tout i tel que descrtab[i].s=1 alors fichier est sauvé)

Traitement.

scannage de la table des descripteurs pour sauver ou détruire les fichiers qui doivent l'être.

Pseudo-code.

```
begin
        receive DESCRTAB from edition-ph-2;
        while (k<21)
            begin
               if DESCRTAB [k] = 1
                 begin
                    link(DESCRTAB.[k].tmp.identif,
                         DESCRTAB [k] identif);
                    remove (DESCRTAB[k].tmp.identif);
                 end
               if DESCRTAB[k].d = 1
                 begin
                   remove(DESCRTAB[k].tmp.identif);
                   remove(DESCRTAB[k].identif);
                 end
               if (DESCRTAB[k].d = 0) et (DESCRTAB[k].g = 0)
                 begin
                   message ('31');
                   get cla;
                   if (get cla = 0)
                          then
                          begin
                            link (DESCRTAB[k].tmp.identif,
                                      DESCRTAB[k].identif);
                            remove (DESCRTAB[k].tmp.identif);
                          end
                          else
                          begin
                            remove (DESCRTAB[k].tmp.identif);
                          end
                 end
end
```

Module: Justifie-fct-2

Entrée.

Document édité,
RENS-FEN,
Numéro de la ligne courante.

Sortie.

Document édité,
RENS-FEN,
Numéro de la ligne courante.
Document ancienne version

Dynamique.

Cette fonction est déclenchée soit par le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectifs :
Cette fonction permet de justifier un document ou une partie d'un document selon les caractéristiques définies dans format-fct-2.

Précondition.

[(il existe un document courant) AND (il existe Rens-fen courant) AND ((choix-util-2 = Création) AND (choix-util-2-c = Justifie)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Justifie))]

Postcondition.

[(il existe un document courant) AND (il existe Rens-fen courant) AND ((choix-util-2 = Création) AND (choix-util-2-c = Justifie)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Justifie)) AND ((texte justifie si possibilité créer temporaire) OR (erreur si pas possibilité))]

Traitement.

- composer un buffer c'est-à-dire lire le document et s'arrêter quand le buffer est rempli (MAXLINE).
- lorsque celui-ci est composé,
  Si justifie à gauche
    - reculer jusqu'au premier blanc
    - insérer un EOL
    - recopier la ligne
    - fin de la justification
  sinon - justifier à gauche et à droite.
  k=MAXLINE;
  Si buffer[k] = " "

```
                        - boucler sur ce test
                        - décrémenter k
                Si buffer[k+1] = espace
                   - calculer le nombre de blanc à ajouter à
                     la ligne
                   - Si le nombre de blanc à ajouter = 0
                        la ligne est justifiée.
                        sinon calculer le nombre de
                        pseudo-blancs à ajouter par blanc.
                   - Si le nombre de pseudo-blanc + 1 > nombre de
                        pseudo-blanc accepté par l'utilisateur
                        couper le mot c-à-d
                        Si coupe = non
                          - reculer jusqu'à avoir un blanc
                          - justifier la ligne sans couper
                        sinon couper le mot
                        Si coupure incorrecte
                     k = adresse du debut du mot + longueur de coupe
                        alors reculer jusqu'à un blanc et
                                 justifier sans couper.
                              sinon insérer les pseudo-blanc.
                        - insérer un pseudo-CR (EOL)
                        - recopier la ligne
                Reculer jusqu'à avoir un blanc suivant une lettre
                et recommencer le même procédé que si on avait
                eu cela du premier coup.
```

Pseudo-code.

```
begin
        receive (fichier, RENS_FEN) from gestion des primitives;
        sauter enreg ENTETE, FORMAT;
        while (read(fichier) != EOF)
                begin
                        i=0;
                        while (i<100)
                        begin
                                z_trav[j] = z_entree[i];
                                if (j = jmax)
                                        begin
                                        j = j - justline(z_trav);
                                        end
                        end
                end
        while (j>1)
                begin
                j = j - justline(z_trav);
                end
        mettre anc_vers du fichier dans copy_und
        et la nouvelle a la place de l'ancienne;
end
```

Module: Justline-fct-2

Pseudo-code

```
 begin

        receive (z_trav) from justifie;
        while (i <= Z_FMT.largeurtxt)
                begin

                   if (CR ou FF) sortir de la boucle;
                   if (TAB) avancer sans justifier jusqu'a
                         la fin de la ligne;
                   if (SAJEX ou EOL ou TIRET) compacter z_trav;
                   ++i;

                end
        reculer jusqu'a un blanc;
        if (Z_FMT.justifg = oui)
                begin

                        if (z_trav[i] != CR ou FF)
                                begin
                                        inserer un EOL en i;
                                end
                        pour j=0;j<=i;++j ecrire z_trav[j];
                        return

                end

 posit: se positionner sur le 1er caractère précédant le blanc;
        if (i = 0)
                begin
                        ecrire la ligne de blanc;
                end
        if (z_trav[i+1] != blanc)
                begin

                        if (Z_FMT.coupe = oui)
                                begin
                                        inserer un pseudo-blanc;
                                end
                        else
                                begin
                                        reculer;
                                        goto posit;
                                end

                end

        calculer le nombre de blancs dans la ligne;
        nombre de SAJEX=long_ligne_max - long_reelle;
        if (nombre blancs = 0)
                begin
                        inserer un EOL;
```

```
                        ecrire;
                end

        calcul du nombre de SAJEX par espace
                SAJSPA = SAJEX / SPACE;
                reste = SAJEX - SAJSPA * SPACE;

        if (SAJSPA > Z_FMT.nb_blanc)
                begin
                        ecrire;
                        return;
                end
        repartir les SAJEX sur la ligne;
end
```

Module: Pagine-fct-2

Entrée.

Document édité
RENS-FEN
Numéro ligne courante
CURSOR

Sortie.

Document édité
RENS-FEN
Numéro ligne courante
CURSOR
Document ancienne version

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectifs :
Cette fonction permet de numéroter toutes les pages d'un document.Elle renumérote aussi les pages du document qui l'étaient déjà lorsque il y a eu des modifications qui ont perturbé celui-ci.
Cette numérotation n'apparait qu'à l'impression.

Précondition.

[(il existe document courant) AND (il existe Rens-fen courant) AND ((choix-uitl-2 = Création) AND (choix-util-2c = Pagine)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Pagine))]

Postcondition.

[(il existe document courant) AND (il existe Rens-fen courant) AND ((choix-util-2 = Création) AND (choix-util-2c = Pagine)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Pagine)) AND (Pagine is True)]

Traitement.

Si dernier caractère de la ligne = caractère de saut de page (FF)
- initialiser le nombre de lignes à 0
- insérer le numéro de page
- incrémenter le numéro de page

Si le nombre de lignes = MAXLINE
- insérer un pseudo-saut de page (EOP)

- remettre le nombre de lignes à 0
- insérer le numéro de page

Sinon incrémenter le nombre de lignes.

Cette pagination est effectuée lors de l'impression sur feuille.

Pseudo-code.

```
begin
        receive (nom-fichier) from gestion des primitives;
        Pagine.nom fichier = True;
end.
```

Module: Vis-car-spec-fct-2

Entrée.

Document édité
RENS-FEN
Numéro ligne courante
CURSOR

Sortie.

Document édité
RENS-FEN
Numéro ligne courante
CURSOR

Dynamique.

Cette fonction est déclenchée par soit le message Choix- msg-2-c soit par le message Choix-msg-2-m.

Description.

Objectifs : Cette fonction permet de voir les caractères spéciaux de formatage. Ces caractères sont essentiellement des pseudo-blancs, des cadratins, des pseudos-CR, des CR, des tirets (ajoutés par la césure), des FF, des pseudo-FF, etc.... Ils ne sont visibles qu'à la demande. Il suffit de positionner un SW qui détermine pour les routines d'affichage que visualiser.

Précondition.

(il existe document courant) AND (il existe Rens-fen courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = Vis. Car.Spec.)) OR ((choix-util-2 = Modification) AND (choix-util-2-c = Vis.Car.Spec.))]

Postcondition.

[(il existe document courant) AND (il existe Rens-fen courant) AND ((choix-util-2 = Création) AND (choix-util-2-c = Vis.Car.Spec.)) OR ((choix-util-2 = Modification) AND (choix-util-2-c = Vis.Car.Spec.)) AND (si caractère = CR afficher <-) OR (si caractère = EOL afficher 1)]

Traitement.

Si le caractère = CR afficher <-' (code ascii 15)
Si le caractère = EOL afficher ¶ (code ascii 12)
Si le caractère = cadratin afficher   (code ascii 177)
Si le caractère = FF   afficher $ (code ascii 14)
Si le caractère = EOP afficher) (code ascii 7)
Si le caractère = sajex afficher .. (code ascii 0)
Si le caractère = tiret afficher } (code ascii 4)
Si le caractère est différent des précédents
                                afficher le caractère.

Pseudo-code.

```
begin
       receive (fichier) from gestion des primitives;
       positionnement du Sw-vis = ON;
end
```

Module: Saut-de-page-fct-2

Entrée.

Document édité
RENS-FEN
Numéro ligne courante
CURSOR

Sortie.

Document édité
RENS-FEN
Numéro ligne courante
CURSOR
Document ancienne version

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectifs :
Cette fonction permet de forcer un saut de page là où on mettra le curseur.

Règles :
Insérer un FF à l'endroit déterminé par l'utilisateur

Précondition.

(il existe document courant) et (il existe Rens-fen courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = Saut de page)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Saut de page))]

Postcondition.

(il existe document courant) et (il existe Rens-fen courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = Saut de page)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Saut de page)) AND (ligne (Pos.cur) = FF) et fenêtre effacée.

Traitement.

Il suffit d'insérer un saut de page à l'endroit où se trouve le curseur.

Pseudo-code.

```
begin
        receive (fichier, CURSOR, RENS_FEN, Numéro de ligne
        courante) from gestion des primitives;
        copier le fichier sur ancienne version;
        begin
           while (getcla != CR) do
                begin
                        déplacement de curseur;
                end
        end
        insert (FF);
        clear-fenêtre;
 end
```

Module: Montre-fct-2

Entrè.

RENS-FEN
Document èditè,
Buffer(n).

Sortie.

RENS-FEN
Document èditè,
Buffer(n).

Dynamique.

Cette fonction est dèclenchèe par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectifs :
Cette fonction permet de voir le contenu d'un des 10 buffers.

Précondition.

(il existe un document courant) AND (il existe Rens-fen courant) AND [((choix-util-2 = Crèation) AND (choix-util-2-c = Montre)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Montre))]

Postcondition.

(il existe un document courant) AND (il existe Rens-fen courant) AND [((choix-util-2 = Crèation) AND (choix-util-2-c = Montre)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Montre))]

Traitement.

ce buffer existe	Y	N
effacer la fenêtre	X	
imprimer le contenu du buffer dans la fenêtre courante	X	
primitives -insertion	X	
-dèplacement		
-delete, BS		
erreur		X
restaurer le contenu de la fenêtre	X	

Cette fonction permet de voir le contenu d'un des 10 buffers.

Il faut acquérir le numéro du buffer. Ensuite, afficher dans la fenêtre courante le contenu du buffer(n). Si l'utilisateur le désire, il peut effectuer sur ce buffer les mêmes primitives que sur un document normal(gestion des primitives). Ensuite, il faut restaurer la fenêtre avec le contenu du document courant.

Pseudo-code.

```
begin
        reveive (fichier,RENS_FEN) from gestion des primitives;
        acquisition du numéro du buffer;
        vérification de l'existence du buffer;
        effacer la fenêtre;
        transférer le contenu de la mémoire buffer (n) dans la
        fenêtre;
        while (getcla n'est pas égal à CR) do
          begin
                primitives de déplacements;
                primitives d'insertion;
                primitives delete;
          end
        affichage ECRAN;
end
```

Module: Copie-fct-2

Entrée.

RENS-FEN
Document édité,
Buffer(n).
CURSOR

Sortie.

RENS-FEN
Document édité,
Buffer(n).
CURSOR
Document ancienne version
Délimiter_texte

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectifs :
Cette fonction permet de copier dans le buffer no Numero- buffer-util-2 le texte delimité par l'utilisateur.

Précondition.

(il existe un document courant) AND (il existe Rens-fen courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = Copie)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Copie))]

Postcondition.

(il existe un document courant) AND (il existe Rens-fen courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = Copie)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Copie))]

Traitement.

buffer existe	Y	Y	N	N
curseur dans texte	Y	N	Y	N
copier	X	X	X	X
délimiter texte à copier	X	X	X	X
amener curseur dans texte courant		X	X	X
erreur				
créer buffer			X	X

- Si c'est un numéro de buffer, acquérir
  le numéro de buffer ou le nom document
- délimiter le texte à copier
- copier le buffer à l'endroit où est le curseur
  dans le document courant.
- si c'est un nom de document
- vérifier la validité de ce nom de document
- délimiter le texte à copier
- copier dans le buffer.

Pseudo-code.

```
begin
        receive (fichier) from gestion des primitives;
        copier le fichier sur ancienne version;
        positionner le curseur dans l'entête;
        afficher : QUOI ?;
        positionner le curseur au début de la fenêtre;
        while get-cla n'est pas égal à ret do
              call déplacement;
        while get-cla n'est pas égal à ret do
            begin
                copie ligne dans buffer;
            end
end
```

Module: Recherche-fct-2

Entrée.

RENS-FEN
Document édité
CHAINE-RECH

Sortie.

RENS-FEN
Fenêtre document

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectifs :
Cette fonction permet de rechercher un string de caractères dans tout le texte.

Précondition.

(il existe RENS_FEN courant) AND (il existe un document courant) AND ((choix-util-2 = Création) AND (choix-util-2-C = Recherche)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Recherche))]

Postcondition.

(il existe RENS_FEN courant) AND (il existe un document courant) AND ((chaine trouvee) OR (chaine pas trouvee)) AND ((choix-util-2 = Création) AND (choix-util-2-C = Recherche)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Recherche))]

Traitement.

Règles :

- acquérir la chaine de caractères à rechercher et le champ de la recherche
- Si la chaine est vide, la recherche est finie
- Si le champ de la recherche est le début du texte, se positionner au début du texte.
- rechercher la chaine dans la ligne courante c-à-d comparer chaque caractère de la chaine avec chaque caractère de la ligne
- Si on a trouvé
  - afficher la ligne
  - positionner le curseur sur ce mot.
- S'il y a déjà eu des caractères trouvés
  si l'on se trouve à la fin de la ligne et que le

dernier caractère de la ligne est un pseudo-tiret ajouté par la justification,
- continuer la recherche sur la ligne suivante.
- incrémenter le numéro de ligne courante.

- sinon
  - rechercher dans la ligne suivante
  - incrémenter le numéro de ligne courante.

fin fichier	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
chaine	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
rech = vide																
chaine rech = chaine trouve	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
lect.clavier = CR	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
impossible	X	X	X	X					X	X						
setcurs					X	X							X	X		
déb. niet,lig																
afficher					X	X										
"NON TROUVE"							X	X								
"TROUVE"					X	X							X	X		
continuer à chercher														X	X	X
fin	X	X	X	X	X	X	X	X			X	X	X			

Cette table peut être compactée car plusieurs fois certaines conditions n'influent pas sur les actions.

fin fichier	-	Y	Y	N	N	N	N
chaine rech = vide	Y	N	N	Y	N	N	N
chaine rech = chaine trouvée	-	Y	N	N	Y	Y	N
lect. clavier = CR	-	-	-	-	Y	N	-
impossible	X						
positeurs début mot, lig		X			X	X	
afficher		X			X	X	
"NON TROUVE"			X				
"TROUVE"		X			X	X	
continuer à chercher						X	X
fin	X	X	X	X	X		

Pseudo-code.

```
begin
        receive(RENS_FEN, document edité) from gestion des
        primitives;
        positionner le curseur dans l'entête;
        display "QUOI ?";
        get chaine à chercher;
        se positionner au début du texte;
lecture:get car;
        if fin de txt   then erreur (5);
                             stop;
        i = 1;
        while i < ou = long. chaine do
                begin
                if car. = chaine (i)    then i = i + 1;
                                        else go to lect.;
                get car.;
                end
        déplacer le texte pour avoir cette ligne sur l'écran
        le curseur au début du mot; (reculer de i +1 lettre);
ordre cla : get cla;
        if get cla = ret then return;
        if get cla = curseur then go to lect;
                             else erreur (6);
                                  go to ordre cla;
end
```

Module: Change-fct-2

Entrée.

RENS-FEN
CURSOR
CHAINE-RECH
CHAINE-SUBST.
Document édité

sortie.

Fenêtre document
Document édité
Document ancienne version
RENS-FEN
CURSOR

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectifs :
Cette fonction permet de changer un string de caractère par un autre.

Règles :
Les règles de recherche sont les mêmes que pour la recherche d'une chaine.
Quand on a trouvé une chaine,

- Si l'utilisateur entre un CR
  - changer la chaine c'est-à-dire,
    si longueur de chaine à changer = longueur de chaine de rechange
    on change la chaine.
    si longueur de chaine à changer < longueur de chaine de rechange
    - diff=long_ch_change-long_ch_rech
    - décaler de diff positions vers la droite
  - insérer la chaine de rechange.
    si longueur de chaine à changer > longueur de chaine de rechange
    - diff=longchainechange-longchainerech
    - décaler de diff positions vers la gauche
    - insérer la chaine de rechange.
- Si l'utilisateur entre Depl-4-bas,
  rechercher l'occurrence suivante de la chaine
- Si l'utilisateur entre un caractère différent,
  ne rien faire.

Précondition.

(il existe RENS_FEN courant) AND (il existe un document courant) AND [((choix-util-2 = Création) AND (choix-util-2-C = Change)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Change))]

Postcondition.

(il existe RENS_FEN courant) AND (il existe un document courant) AND [((chaine trouvée) OR (chaine pas trouvée)) AND [((choix-util-2 = Création) AND (choix-util-2-C = Change)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = Change))].

Traitement.

Le traitement est le même que dans le cas de la fonction de Recherche-fct mais dans le cas ou on a TROUVE, il faut substituer les deux chaînes. La substitution de ces deux chaînes est expliquée dans la table suivante.

long. chaine rech = long. chaine sub.	Y	Y	Y	Y	N	N	N	N
long. chaine rech > long. chaine sub.	Y	Y	N	N	Y	Y	N	N
long. chaine rech < long. chaine sub.	Y	N	Y	N	Y	N	Y	N
impossible	X	X	X		X			X
décaler le texte à partir de déb-mot vers la droite de la différence entre long chaine rech. et long chaine sub.							X	
décaler le texte à partir de déb-mot vers la gauche de la différence entre long chaine rech. et long chaine sub.						X		
substituer les deux chaines				X		X	X	

Pseudo-code.

```
begin
        receive (RENS_FEN,CURSOR,document édité) from gestion des
        primitives;
        copie du fichier dans ancienne version;
        positionner le curseur;
        afficher "QUOI ?";
        get chaine à changer ;
        afficher "PAR ?";
        get chaine de changement;
        se positionner au début du texte;
        lecture : getcar;
                if fin de texte       then erreur (5);
                                        stop;
                i = 1;
                while i < long. chaine do
                        begin
                                if car = chaine (i)
                                        then i=i+1;
                                        else go to lect;
                                get car;
                        end
        déplacer le texte pour avoir cette ligne sur l'écran;
        ordre cla : get cla;
        if change = get cla then substituer les deux chaines;
        if get cla = ret then stop;
        if get cla = cur then go to lect;
                        else erreur (6);
                        go to ordre cla;
end
```

Module: Minuscule-fct-2

Entrée.

Document édité
CURSOR
RENS-FEN
Numéro ligne courante

Sortie.

Document édité
RENS-FEN
CURSOR
Document ancienne version

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectifs :
Cette fonction permet de changer en minuscule,si nécessaire toutes les lettres qui passent sous le curseur.

Précondition.

(il existe Rens-fen courant) AND (il existe un document courant) AND [((choix-util-2 = Création) AND (choix-util-2-C = Minuscule)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Minuscule))]

Postcondition.

(il existe Rens-fen courant) AND (il existe un document courant) AND AND [((choix-util-2 = Création) AND (choix-util-2-C = Minuscule)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Minuscule))]

Traitement.

Il faut d'abord délimiter le texte à mettre en minuscule. Ensuite, pour chaque caractère de la portion de texte délimitée si 101 (octal) <= caractère <= 132 (octal) alors additionner 40 (octal) Sinon ne rien faire.

Pseudo-code.

```
begin
        receive (fichier) from gestion des primitives;
        copie du fichier dans ancienne version;
        c = get cla;
        if (c = dépl-déb-txt)
                then initialiser au début du texte;
        délimiter texte (nom-document);
        while not fin portion cité do
                begin
                        if car-lu appartient à {41,--5A}
                                car-lu = car-lu + 20 (hexa);
                        avancer dans le texte;
                end
 end
```

Module: Majuscule-fct-2

Entrée.

Document edité
CURSOR
RENS-FEN
Numéro ligne courante

Sortie.

Document edité
CURSOR
RENS-FEN
Document ancienne version

Dynamique.

Cette fonction est déclenchée par soit le message Choix-util-2-C soit par le message Choix-util-2-M.

Description.

Objectifs :
Cette fonction permet de changer en majuscule,si nécessaire toutes les lettres qui passent sous le curseur.

Précondition.

(il existe RENS_FEN courant) AND (il existe un document courant) AND [((choix-util-2 = Création) AND (choix-util-2-C = Majuscule)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Majuscule))]

Postcondition.

(il existe RENS_FEN courant) AND (il existe un document courant) [((choix-util-2 = Création) AND (choix-util-2-C = Majuscule)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = Majuscule))].

Traitement.

Il faut d'abord délimiter le texte à transformer en majuscule.
Ensuite, pour chaque caractère de la portion de texte délimité
si 141(octal)<=caractère<=172(octal)
alors soustraire 40(octal).
sinon ne rien faire.

Pseudo-code.

```
begin
        receive (fichier)from gestion des primitives;
        copie  du fichier dans ancienne version;
        c = get cla;
        if (c = dépl-déb-txt)
                then initialiser au début du texte;
        délimiter texte (nom-document);
        while not fin portion do
                begin
                        if car-lu appartient à {41,--5A}
                                car-lu = car-lu - 20 (hexa);
                        avancer dans le texte;
                end
end
```

Module: Transfert-fct-2

Entrée.

Document édité
CURSOR
Numéro ligne courante
Numéro ligne début
Numéro ligne fin
Numéro caractère début
Numéro caractère fin
Numéro ligne de transfert

sortie.

Document édité
Document ancienne version
CURSOR

Sous-partie.

délimiter texte

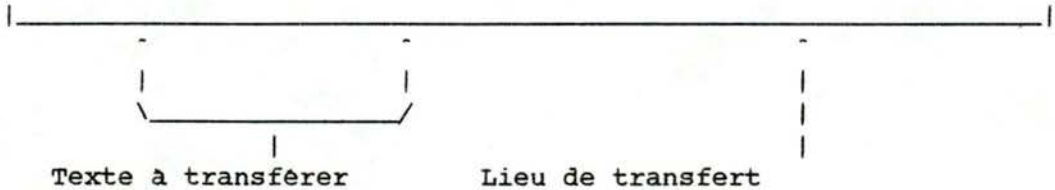
Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

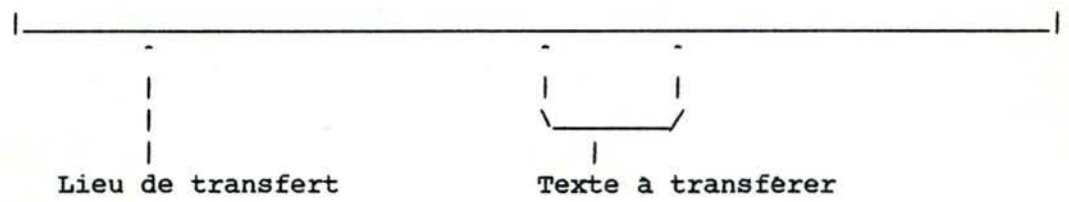
Description.

Objectifs :
Cette fonction permet de transférer à l'intérieur d'un document ou même d'un document à un autre une portion de texte délimité par l'utilisateur.

Règles:
Il faut d'abord délimiter le texte à transférer
par la fonction Délimiter_texte.
Ensuite, il faut déterminer l'endroit où transferer cette portion de texte.
Si l'utilisateur entre au clavier un déplacement
alors effectuer ce déplacement.
Sinon no-ligne de transfert=no-ligne-courante.
Copier la portion de texte dans une zone temporaire.
Si la ligne de transfert est > que le numéro
de la ligne du début du texte a transférer
- Décaler ce qui suit la portion de texte
vers le début du texte.
- Insérer la zone temporaire à sa place.



Si la ligne de transfert est < que le numéro
de la ligne du début du texte à transférer
- Décaler le texte qui suit la portion
de texte vers ce qui suit cette portion.
- Insérer la zone temporaire à sa place.



Précondition.

(il existe Rens-fen courant) AND (il existe un document courant)
{((choix-util-2 = Création) AND (choix-util-2-C = Transfert)) OR
((choix-util-2 = Modification) AND (choix-util-2-M = Transfert))]

Postcondition.

(il existe Rens-fen courant) AND (il existe un document courant)
AND [((choix-util-2 = Création) AND (choix-util-2-C = Transfert))
OR ((choix-util-2 = Modification) AND (choix-util-2-M =
Transfert))]

Traitement.

Mess op = numéro du buffer	Y	Y	Y	Y	N	N	N	N
il existe un buffer	Y	Y	N	N	Y	Y	N	N
curseur dans texte	Y	N	Y	N	Y	N	Y	N
posit. curseur dans texte		X				X		X
transfert dans buffer					X	X	X	X
délimiter texte à copier					X	X	X	X
créer buffer								
transfert dans texte	X	X			X	X	X	X
erreur			X	X				

Cette primitive permet 2 types d'opérations de copiage :
- elle permet le transfert du texte d'un endroit dans un autre de celui-ci.
- elle permet de copier le contenu d'un buffer n'appartenant pas au même document.

Pseudo-code.

```
begin
        receive (fichier) from gestion des primitives;
        copier le fichier dans ancienne version;
        positionner le curseur dans l'entête;
        afficher QUOI ?;
        get cla;
        while get cla n'est pas égal ret do
             begin
                déplacement;
                get cla;
            end
        if get cla = numéro du buffer go to  OU;
        mémoriser la position;
        get cla;
        while get cla n'est pas égla ret do
            begin
                déplacement;
                get cla;
            end
        mémoriser la position;
        positionner le curseur dans l'entête;
OU:     afficher OU ? ;
        get cla;
        while get cla n'est pas égal à ret do
                déplacement;
                [get cla];
        mémoriser la position : lieu de transfert;
        transférer le texte à déplacer dans une zone;
        recopier le texte suivant le texte à déplacer
        sur le texte à déplacer jusqu'à l'endroit
        où se trouve la position de transfert;
        copier le buffer à partir de la position de
        transfert;
end
```

Module: Caractère-fct-2

Entrée.

Document édité
RENS-FEN
CURSOR
Numéro ligne courante

Sortie.

Document édité
RENS-FEN
CURSOR
Numéro ligne courante
Document ancienne version

Sous-partie.

Délimiter_texte

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m. Elle reçoit le message Caractère-util-2. La fonction caractère génère le menu-caractère-2-*

Description.

Objectif:
Cette fonction permet de choisir certaines caractéristiques pour la partie de texte délimitée par l'utilisateur.

Règles:
Il faut d'abord délimiter le texte à transformer. Ensuite, après que l'utilisateur aie choisi le style de caractère pour cette portion de texte (Messenger,Italique, Helvesan,Grec,Souligne,Gras,Surligne), il suffit d'apposer le masque adéquat sur chaque caractère de la portion de texte.

Précondition.

(il existe document-courant) AND (il existe RENS_FEN courant) AND [((choix-util-2=Création) AND (choix-util-2-c=Caractere)) OR ((choix-util-2=Modification) AND (choix-util-2-m=Caractere))]

Postcondition.

(il existe document-courant) AND (il existe RENS_FEN courant) AND [((choix-util-2=Création) AND (choix-util-2-c=Caractere)) OR ((choix-util-2=Modification) AND (choix-util-2-m=Caractere))] AND (choix-util appartient à {GR, HT, GS, SO, SU, IT, GC, MS}

Traitement

Il faut d'abord délimiter le texte à dont il faut changer le style.

choix utilisateur	GR	HE	GS	SO	SU	IT	GC	MS
style = GRAS	X							
style = HELVESAN		X						
style = GRAS SOUL			X					
style = SOUL				X				
style = SURL					X			
style = ITAL						X		
style = GREC							X	
style = MESSENGER								X

Pseudo-code.

```
begin
        receive (fichier, rens-fen) from gestion des
        primitives;
        copier le fichier sur 2ème version;
        c = get clavier;
        while (c != CR) do
                begin
                        déplacement curseur;
                end
        read (choix);
                begin
                        case "GR" : STYLE = "GR";
                        case "HE" : STYLE = "HE";
                        case "GS" : STYLE = "GS";
                        case "SO" : STYLE = "SO";
                        case "SU" : STYLE = "SU";
                        case "IT" : STYLE = "IT";
                        case "GC" : STYLE = "GC";
                        case "MS" : STYLE = "MS";
                end
end
```

Module: Encadre-fct-2

Entrée.

Document édité
RENS-FEN
CURSOR
Numéro ligne courante

Sortie.

Document édité
RENS-FEN
CURSOR
Document ancienne version

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectif:
Cette fonction permet d'encadrer tout ou une partie quelconque d'un document.

Précondition.

(il existe Rens-fen courant) AND (il existe un document courant) AND ((choix-util-2 = Création) AND (choix-util-2-C = Encadre)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Encadre))]

Postcondition.

(il existe Rens-fen courant) AND (il existe un document courant) AND [((choix-util-2 = Création) AND (choix-util-2-C = Recherche)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Recherche))]

Traitement.

L'utilisateur détermine le coin supérieur gauche et le coin inférieur droit de l'encadrement en se positionnant sur le 1er caractère du texte à encadrer et sur le dernier caractère du texte à encadrer. Ces 2 positions sont mémorisées. Ensuite, il faut ajouter à la matrice représentant chaque caractère

- de la ligne supérieure un masque de facon à surligner le caractère
- commencant chaque ligne du texte à enc encadrer un ligne verticale avant le caractère
- teminant chaque ligne du texte à encadrer un ligne verticale après le caratère

- de la ligne inférieure un masque de facon à souligner le caractère

On obtient donc un encadrement qui est intégré aux bornes du texte à encadrer.

Pseudo-code.

```
begin
        receive (fichier,RENS_FEN) from gestion des
        primitives;
        delimiter texte (fichier, RENS_FEN);
           on recoit une donnee de type FENETRE : CADRE
                        cadre.coordx1   no_car
                        cadre.coordy1   no_lig
                        cadre.coordx2   no_car
                        cadre.coordy2   no_lig
        masquer le caractere en y1 x1 par |-;
        masquer le caractere en y1 x2 par -|;
        masquer le caractere en y2 x1 par L;
        masquer le caractere en y2 x2 par _|;
        masquer les caracteres de y1 x1+1 -> y1 x2-1
        caractere;
        masquer les caracteres de y2 x1+1 -> y2 x2-1
        caractere;
        masquer les caracteres de y1+1 x1 -> y2-1 x1
        |caractere;
        masquer les caracteres de y1+1 x2 -> y2-1 x2
        caractere|;
end
```

Module: Defaire-fct-2

Entrée.

Document édité à l'étape T
Document ancienne version

Sortie.

Document édité à l'étape T - 1
Document ancienne version

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectif:
Cette fonction permet d'annuler l'effet de la commande précédente.

Précondition.

(il existe RENS_FEN courant) AND (il existe un document courant) AND [((choix-util-2 = Création) AND (choix-util-2-C = Défaire)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Défaire))]

Postcondition.

(il existe RENS_FEN courant) AND (il existe un document courant) AND [((choix-util-2 = Création) AND (choix-util-2-C = Défaire)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = Défaire))].

Traitement.

Il suffit de repartir de l'ancienne version du document qui a été conservée après chaque opération effectuée sur le document.

Pseudo-code.

```
begin
        receive (fichier) from gestion des primitives;
        rename (ancienne version, intermédiaire);
        rename (fichier, ancienne version);
        remove (fichier);
        rename (intermédiaire, fichier);
        remove (intermédiaire);
end
```

Module: Efface-fct-2

Entrée.

Document édité
RENS-FEN
CURSOR
Numéro ligne courante

Sortie.

Document édité
Document ancienne version
RENS-FEN
CURSOR
Numéro ligne début
Numéro ligne fin
Numéro caractère début
Numéro caractère fin
Numéro ligne courante

Sous-partie.

Délimiter_texte

Dynamique.

Cette fonction est déclenchée par soit le message Choix-util-2-c soit par le message Choix-util-2-m.

Description.

Objectif:
Cette fonction permet d'effacer tout ce qui se trouve entre deux bornes définies par l'utilisateur.

Traitement.

Il faut d'abord délimiter la partie de texte à effacer en mémorisant les bornes sous la forme x1,y1 et x2,y2. Ensuite,

y1=y2	N	N	N	N	Y	Y	Y	Y
x1=debut ou fin ligne	N	N	Y	Y	N	N	Y	Y
x2=debut ou fin ligne	N	Y	N	Y	N	Y	N	Y
detruire ligne y1->y2			X					X
detruire ligne y1->y2-1				X				
decaler y2,x2+1->y1,x1					X	X	X	
recopie ligne y1					X	X	X	
decaler y2,x2+1->y2,0	X		X					
effacer apres y1,x1	X	X						
detruire lig y1+1->y2		X						
detruire lig y1+1->y2-1	X							
afficher l'ecran	X	X	X	X	X	X	X	X

Pseudo-code

```
begin
        receive (fichier, RENS_FEN, CURSOR) from gestion
        des primitives;
        delimiter texte (fichier, RENS_FEN, CURSOR);
        read_rand(fichier,cadre.coordy1,no_lig_cour,BUFSIZE);
        if (cadre.coordy2 = cadre.coordy1)
                begin
                        compacter la ligne courante;
                        ecrire;
                        return;
                end
        for (i = cadre.coordy1, i<cadre.coordy2, ++i)
                begin
                        effacer la ligne i;
                end
        compacter les lignes cadre.coordy1 et cadre.coordy2
        en une seule ligne;
end
```

Module: Centre-fct-2

Entrée.

Document édité
CURSOR
RENS-FEN
Numéro ligne courante

Sortie.

Document édité
Document ancienne version
Numéro ligne courante
RENS-FEN
CURSOR

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectif:
Cette fonction permet de centrer le texte se trouvant sur la ligne pointée par le curseur.

Précondition.

(il existe RENS_FEN courant) AND (il existe un document courant) AND [((choix-util-2 = Création) AND (choix-util-2-C = Centre)) OR ((choix-util-2 = Modification) AND (choix-util-2-M = Centre))]

Postcondition.

(il existe RENS_FEN courant) AND (il existe un document courant)
AND [((choix-util-2 = Création) AND (choix-util-2-C = Centre)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = Centre))].

Traitement.

Calculer le numéro de la ligne a centrer.
Calculer la longueur du string à centrer ainsi que l'adresse du début du string.
Calculer le nombre de blancs à inserer avant le string
Si le nombre de blancs à insérer = l'adresse du début du string, il n'y a rien à faire.
Si le nombre de blancs à insérer < l'adresse du début du string, décaler le string de N positions vers la gauche.
N=adresse début-nombre de blancs à insérer.
Si le nombre de blanc à insérer > l'adresse du début

du string, décaler le string de N positions vers la droite.
N=nombre de blancs à insérer-adresse début.

Pseudo-code.

```
begin
   receive (nom-document, numéro lig cour) from gestion
   des primitives;
   calcul de la longueur du string à centrer = longstring;
   calcul du nombre de blancs avant le string : début;
   calcul du nombre de blancs à mettre avant le string pour
               le centrer = nombre de blancs;
   if nombre de blancs < ou = début
                    décaler vers la gauche le
                    string jusque le nombre de blancs + 1;
   if nombre de blancs > début
                      décaler vers la droite le string
                      jusque le nombre de blancs + 1;
end
```

Module: Del-fct-2

Entrée.

Document édité
ECRAN
CURSOR
Numéro ligne courante

Sortie.

ECRAN
Document édité
CURSOR
Numéro ligne courante

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectif:
Cette fonction permet de détruire le caractère qui se trouve sous le curseur.

Règles:
Il suffit de décaler les caractères suivant celui où positionné le curseur de une position vers la gauche. Si on veut effacer un CR,LF,EOP,... le procédé est le même et c'est la justification qui se charge de remettre le texte sous un forme correcte.

Précondition.

(il existe un document courant) et (il existe une fenêtre-courant)

Postcondition.

(il existe un document courant) et (il existe une fenêtre courant) et (longueur ligne = longueur ligne - 1 si caractère est différent de CR, FF, EOL, TIRET) et (si caractère = CR, FF, EOL, TIRET et si longueur ligne > 1, longueur [ligne] = longueur [ligne] + nombre de caractère cmplétant ligne + 1, longueur [ligne + 1] = longueur [ligne + 1] - nombre de caractère complétant ligne) et (si caractère = CR, FF, EOL, TIRET et si longueur ligne < ou = 1 le nombre ligne = nombre ligne - 1)

Traitement.

Le traitement consiste à effacer le caractère avant le curseur. Ensuite il faut reculer celui-ci d'une position vers la gauche et de décaler le reste de la ligne vers la gauche. Si le caractère à effacer est un CR, FF, EOL, TIRET, les lignes ne seront réajustées que si l'utilisateur demande une justification.

Pseudo-code.

```
begin
   receive (fichier, RENS_FEN, numéro lig courante,ECRAN);
   getcurs (X, Y);

   calcul de la position du caractère à supprimer

       poscar X = X - bord gauche fenêtre;
       poscar Y = Y - bord supérieur fenêtre + 1;
       numéro lig_deb = numéro lig cour. - poscar Y;

   if écran [poscar Y] [poscar X] = CR, EOL, FF, TIRET
      then -copier les car. de ECRAN [poscar Y + 1] à la
            suite de ECRAN [poscar Y] [poscar X - 1]
            jusque CR, EOL, FF, ou ((poscar X > MAXLINE)
            et (ECRAN [poscar Y] [poscar X] = blancs))
           -effacer un enregistrement(numéro lig cour + 1)
            si (CR, EOL, FF = ECRAN [poscar Y] [poscar X]);
           -décaler les car. restant en ECRAN [poscar Y +
            1] vers le début (ECRAN [poscar Y + 1] [0])
           -affiche (fichier, numéro lig-deb, long-fen, 1);
      else -for (i = poscar X, i < Imax - 1, ++ i)
            ECRAN [poscar Y] [i] = ECRAN [poscar Y] [i + 1]
   affiche ECRAN;
end
```

Module: Insertion-fct-2

Entrée.

ECRAN
Caractère à insérer
Document édité
Numéro de ligne courante
CURSOR

Sortie.

ECRAN
Document édité
Numéro de ligne courante
CURSOR

Dynamique.

Cette fonction est déclenchée par soit le message Choix- util-2-c soit par le message Choix-util-2-m.

Description.

Objectif:
Cette fonction insère des caractères a l'endroit ou se le curseur.

Règles:
Si le caractère à insérer = CR ou LF
Inserer une nouvelle ligne dans le document.
Decaler les lignes qui suivent dans l'écran.
Décaler les caractères suivant l'endroit
où le nouveau caractère s'insère
sur la ligne suivante(vide par décalage des
lignes).
Réafficher l'écran.
Si le caractère à insérer est un blanc et que
la position de ce caractère dépasse la largeur du
texte définie dans le FORMAT il faut inserer un LF.
Sinon il suffit de décaler les caractères a partir de
la position du nouveau caractère à insérer de une
position vers la droite et de réafficher l'écran.

Précondition,

(il existe RENS_FEN courant) AND (il existe un fichier courant) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2) AND (RENS_FEN.cordy1 < Y < RENS_FEN.cordy2)

Postcondition.

(il existe RENS_FEN courant) AND (il existe un fichier courant) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2) AND (RENS_FEN.cordy1 < Y < RENS_FEN.cordy2) AND (caract insérer en bonne place selon les règles de la table de décision)

Traitement.

calcul pos Y / pos X																
caract. =	SPA	CR	FF	#	SPA	CR	FF	#	SPA	CR	FF	#	SPA	CR	FF	#
pos X = Z_FMT.lrg_txt	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
pos Y % long_fen - 1	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
décaler pos à 24 en pos Y + 1,... pos Y+24+1	Y	X	X	X		X	X	X			X	X			X	X
décaler pos à 199 en pos X + 1 à pos X + 200	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
insérer car. en posX posY	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
insérer LF posX posY	X				X											
posX=posX+1	X				X											
scroller l'écran vers le haut	X	X	X							X	X					
recopier les lignes modif.	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
mettre car. en posX posY 0,posY+1 à fin	X	X	X		X	X	X			X	X			X	X	

Pseudo-code.

```
begin
        receive (caractère,ECRAN,fichier,RENS-FEN,CURSOR)
        from gestion des primitives;
        /* calcul de la position du caractère */
        pos X = Rens-fen . cord X 1 - col + 1;
        pos Y = Rens-fen . cord Y 11 - lig + 1;
                /* insertion du caractère */
        if caractère = CR ou FF
        begin
          décaler ECRAN [pos Y] [] vers [pos Y + 1] [];
          décaler ECRAN en pos X, pos Y vers 0, pos Y + 1;
          insérer caractère en pos X, pos Y;
          réafficher ECRAN;
              si pos Y = Rens-fen.long-fen-7
                then scroller l'écran vers le haut;
                else réafficher l'écran;
          incrémenter lig,col;
          return;
        end
        if (caractère = " " et pos X >= Z_FMT.largeur_txt)
        begin
           insérer caractère;
           insérer un LF;
           afficher ECRAN;
           return;
        end
        for (i = 200, i > ou = pos X; --i)
             ECRAN [pos Y] [i] = ECRAN [pos Y] [i - 1];
        insérer caractère;
        afficher ECRAN;
end
```

Module: Depl-l-haut-fct-2

Entrée.

RENS_FEN
Numéro de ligne courante
CURSOR
Document édité

Sortie.

Numéro de ligne courante
CURSOR
Document édité

Dynamique.

Cette fonction est déclenchée soit par le message Choix-util-2-c soit par le message Choix-util-2-m soit par le message Choix-util-2-l.

Description.

Objectif:
Cette fonction permet de déplacer le curseur de une ligne vers le haut dans le document affiché à l'écran.

Précondition.

(il existe un document courant) AND (il existe RENS-FEN courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-l-haut)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-l-haut)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-l-haut))] AND (RENS_FEN.cordx1< x < RENS_FEN.cordx2) AND (RENS_FEN.cordy11< y < RENS_FEN.cordy2).

Postcondition.

(il existe un document courant) AND (il existe RENS-FEN courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-l-haut)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-l-haut)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-l-haut))] AND (RENS_FEN.cordx1< x < RENS_FEN.cordx2) AND (RENS_FEN.cordy11< y < RENS_FEN.cordy2) et (NBLIG=NBLIG-1)

Traitement.

Règles:
Si YCURSOR = RENS_FEN.cordyll+1
- recopier la dernière ligne de la fenêtre dans le document.
- effectuer un décalage des lignes de la fenêtre d'une ligne vers le bas.
- afficher la ligne du document précédant la 1ere ligne de la fenêtre.
- no_lig_cour= no_lig_cour - 1

sinon - YCURSOR = YCURSOR - 1
- setcurs(YCURSOR,XCUSOR)
- no_lig_cour=no_lig_cour- 1

Y=RENS_FEN.cordyll	Y	N
scroller la fen. de une ligne vers le bas	X	
positionner le curseur		X

Pseudo-code.

```
begin
   receive (RENS_FEN,CURSOR,fichier) from primitives;
   get position du curseur sous la forme X,Y;
   if (Y = RENS_FEN.cordyll+1)
       then begin
                 pour toute les lignes de l'écran
                                ecran[i+1]=ecran[i];
                 afficher les lignes depuis 0
                 jusque RENS_FEN.lon_fen - 7;
            end
       else setcurs(Y-1,X);
end
```

Module: Depl-l-bas-fct-2

Entrée.

RENS-FEN
Numéro de ligne courante
CURSOR
Document édité
ECRAN

Sortie.

Numéro de ligne courante
CURSOR
Document édité
ECRAN

Dynamique.

Cette fonction est déclenchée soit par le message Choix-util-2-c soit par le message Choix-util-2-m soit par le message Choix-util-2-l.

Description.

Objectif:
Cette fonction permet de déplacer le curseur de une ligne vers le bas dans le document affiche à l'ecran.

Précondition.

(il existe un document courant) AND (il existe RENS-FEN courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-l-bas)) OR ((choix-util-2- = lecture) AND (choix-util-2-L = depl-l-bas)) OR ((choix-util-2 =Modification) AND (choix-util-2-m = depl-l-bas))] AND
(RENS_FEN.cordx1<= x < RENS_FEN.cordx2) et (RENS_FEN.cordy11<= y < RENS_FEN.cordy2)

Postcondition.

(il existe un document courant) AND (il existe RENS-FEN courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-l-bas)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-l-bas)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-l-bas))] AND (RENS_FEN.cordx1< x < RENS_FEN.cordx2) AND (RENS_FEN.cordy11< y < RENS_FEN.cordy2) AND (nblig=nblig+1)

Traitement.

Règles:
Si YCURSOR = RENS_FEN.cordy2-1
alors - recopier dans le document la

```
                                    lere ligne de la fenêtre.
                                  - décaler les lignes de ECRAN
                                    de une ligne vers le haut.
                                  - afficher la ligne suivant la
                                    dernière ligne de ECRAN.
                                  - no_lig_cour=no_lig_cour +1
                             sinon - YCURSOR=YCURSOR +1
                                  - setcurs(YCURSOR,XCURSOR)
                                  - no_lig_cour=no_lig_cour+1
```

Y=RENS_FEN.cordy2	Y	N
scroller la fen. de une ligne vers le haut	X	
positionner le curseur		X

Pseudo-code.

```
begin
    receive (RENS_FEN,ECRAN,CURSOR,fichier) from gestion
    des primitives;
    get position du curseur sous la forme X,Y;
    if (Y = RENS_FEN.cordy2-1)
       then begin
                  pour toute les lignes de l'écran
                                ECRAN[i]=ECRAN[i-1];
                  afficher ECRAN;
            end
       else setcurs(YCURSOR+1,XCURSOR);
end
```

Module: Depl-4-haut-fct-2

Entrée.

ECRAN
Numéro de ligne courante
CURSOR
RENS-FEN
Document édité

Sortie.

ECRAN
Document édité
CURSOR
Numéro de ligne courante

Dynamique.

Cette fonction est déclenchée soit par le message Choix-util-2-c soit par le message Choix-util-2-m soit par le message Choix-util-2-l.

Description.

Cette fonction permet de déplacer le curseur de quatre lignes vers le haut dans la document affiche à l'ecran. Il suffit de faire le même raisonnement que pour le déplacement de 1 vers le haut et de le faire quatre fois.

Précondition..

(il existe un document courant) AND (il existe RENS-FEN courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-4-haut)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-4-haut)) OR ((choix-util-2 = MOdification) AND (choix-util-2-m = depl-4-haut))] AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2) et (RENS_FEN.cordy11 < Y < RENS_FEN.cordy2)

Postcondition.

(il existe un document courant) AND (il existe RENS-FEN courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-4-haut)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-4-haut)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-4-haut))] AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2) et (RENS_FEN.cordy11 < Y < RENS_FEN.cordy2) AND (NBLIG = NBLIG - 4)

Traitement.

4 fois appel à déplacement-1-haut.

Pseudo-code.

```
begin
        receive (RENS-FEN,ECRAN,CURSOR,fichier)
        from gestion des primitives;
        initialisation : i = 0;
        while (i < 4)
                begin send (fichier,RENS_FEN)  to
                      déplacement-1-haut;
                      i = i + 1;
                end end
```

Module: Depl-4-bas-fct-2

Entrée.

ECRAN
RENS-FEN
CURSOR
Document edité
Numéro de ligne courante

Sortie.

ECRAN
Document edité
CURSOR
Numéro de ligne courante

Dynamique.

Cette fonction est déclenchée soit par le message Choix-util-2-c soit par le message Choix-util-2-m soit par le message Choix-util-2-l.

Description.

Cette fonction permet de déplacer le curseur de quatre lignes vers le bas dans le document affiche à l'ecran. Il suffit de faire le même raisonnement que pour le deplacement de 1 vers le bas.

Précondition..

(il existe un document courant) AND (il existe RENS-FEN courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-4-bas)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-4-bas)) OR ((choix-util-2 = MOdification) AND (choix-util-2-m = depl-4-bas))] AND (RENS_FEN.cordx1 < ou = X < ou = RENS_FEN.cordx2) et (RENS_FEN.cordy11 < Y < RENS_FEN.cordy2)

Postcondition.

(il existe un document courant) AND (il existe RENS-FEN courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-4-bas)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-4-bas)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-4-bas))] AND (RENS_FEN.cordx1 < ou = X < ou = RENS_FEN.cordx2) AND (RENS_FEN.cordy11 < Y < RENS_FEN.cordy2) AND (NBLIG = NBLIG + 4)

Traitement.

4 fois appel à déplacement-1-bas.

Pseudo-code :

```
begin
        receive (RENS_FEN,ECRAN,CURSOR,fichier)
        from gestion des primitives;
        initialisation : i = 0;
        while (i < 4)
           begin
                 déplacement-1-bas;
                 i = i + 1;
           end
end
```

Module: Depl-deb-txt-fct-2

Entrée.

RENS-FEN
CURSOR
Numèro de ligne courante
Document èditè

Sortie.

CURSOR
Numèro de ligne courante
Document èditè

Dynamique.

Cette fonction est dèclenchée soit par le message Choix-util-2-c soit par le message Choix-util-2-m soit par le message Choix-util-2-l.

Description.

Objectif:
Cette fonction permet de dèplacer le curseur au dèbut du texte.

Prècondition.

(il existe document courant) AND (il existe RENS_FEN courante) AND [((choix-util-2 = Crèation) AND (choix-util-2-c = depl-deb-txt)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-deb-txt)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-deb-txt))] AND (RENS_FEN.cordy1 < ou = Y < ou = RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < x < RENS_FEN.cordx2).

Postcondition.

(il existe document courant) AND (il existe RENS_FEN courante) AND [((choix-util-2 = Crèation) AND (choix-util-2-c = depl-deb-txt)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-deb-txt)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-deb-txt))] AND (RENS_FEN.cordy1 < ou = Y < ou = RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < x < RENS_FEN.cordx2) AND (NBLIG=0)

Traitement.

numèro enreg. cour = 0	Y	N
secteurs (YMIN, XMIN)	X	X
se positionner au dèbut du fichier		X
initialiser le num. ligne courante= long. fen.	X	X
afficher le document	X	X

Pseudo-code.

```
begin
    receive (fichier,ECRAN,RENS-FEN,CURSOR) from
    gestion des primitives;
    if (numèro enreg. différent de 0)
                        then positionner dèbut fichier;
                             afficher ECRAN;
    setcurs(RENS_FEN.cordyll + 1, RENS_FEN.cordxl + 1);
end;
```

Module: Depl-fin-txt-fct-2

Entrée.

RENS-FEN
CURSOR
Numéro de ligne courante
Document édité
ECRAN

Sortie.

CURSOR
Numéro de ligne courante
ECRAN
Document édité

Dynamique.

Cette fonction est déclenchée soit par le message Choix-util-2-c soit par le message Choix-util-2-m soit par le message Choix-util-2-l.

Description.

Objectif:
Cette fonction permet de déplacer le curseur à la fin du texte.

Précondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND [((choix-util-2 = Création) AND (choix-util-2-c = depl-fin-txt)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-fin-txt)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-fin-txt))] AND (RENS_FEN.cordy11 < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2)

Postcondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND [((choix-util-2- = Création) AND (choix-util-2-c = depl-fin-txt)) OR ((choix-util-2 = lecture) AND (choix-util-2-L = depl-fin-txt)) OR ((choix-util-2 = Modification) AND (choix-util-2-m = depl-fin-txt))] AND (RENS_FEN.cordy11 < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2) AND (NBLIG=Nombre maximum)

Traitement.

numèro enreg. = NOMAX	Y	N
secteurs (YMAX, XMAX)	X	X
se positionner à la fin du num. ligne courante=LINEMAX		X
fichier		
afficher le nombre de ligne égale à la long. fenètre		X

Pseudo-code .

```
begin
   receive (fichier,ECRAN,CURSOR,RENS-FEN) from gestion des
   primitives;
   if (numèro enreg. # NOMAX)
                          then position-fin-fichier;
                               afficher ECRAN;
   setcurs (YMAX,XMAX);
end
```

Module: Depl-car-gau-fct-2

Entrée.

RENS-FEN
CURSOR
Document édité
ECRAN

Sortie.

Document édité
CURSOR
ECRAN

Dynamique.

Cette fonction est déclenchée soit par le message Choix-util-2-c soit par le message Choix-util-2-m soit par le message Choix-util-2-l.

Description.

Objectif:
Cette fonction permet de déplacer le curseur de un caractère vers la gauche.

Précondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND (RENS_FEN.cordyll < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordxl < X < RENS_FEN.cordx2)

Postcondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND (RENS_FEN.cordyll < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordxl < X < RENS_FEN.cordx2) AND (NBCAR=NBCAR-1)

Traitement.

```
Si XCURSOR = RENS_FEN.cordxl+1
alors Si la fenêtre est totalement décalée
        le signaler à l'utilisateur.
      Sinon déplacer le document sous la
        fenêtre d'un caractère vers la
        gauche.
Sinon X=X+1;
      setcurs(Y,X);
```

X = XMIN	Y	N
setcurs (Y, X - 1)		X
scroller vers la drt le document dans la fenêtre	X	
début =	X	

Pseudo-code

```
begin
  receive (ECRAN, RENS_FEN, fichier, CURSOR) from gestion
  des primitives;
  if ( X > RENS_FEN.coordx1 )
       begin
              X = X - 1;
              setcurs(Y,X);
       end
  else
       begin
          afficher en décalant de 1 caractère vers
          la gauche;
       end
end
```

Module: Depl-car-dr-fct-2

Entrée.

Document édité
CURSOR
RENS_FEN
ECRAN

Sortie.

CURSOR
RENS-FEN
ECRAN

Dynamique.

Cette fonction est déclenchée soit par le message Choix-msg-3-c soit par le message Choix-msg-3-m soit par le message Choix-msg-3-l.

Description.

Objectif:
Cette fonction permet de déplacer le curseur de un caractère vers la droite.

Précondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND (RENS_FEN.cordy11 < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2)

Postcondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND (RENS_FEN.cordy11 < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2) AND (NBCAR=NBCAR+1)

Traitement.

Si XCUSOR=RENS_FEN.cordx2-1
alors Si la fenêtre est décalée au maximum
le signaler à l'utilisateur.
Sinon déplacer la fenêtre de 1 caractère
vers la droite.
Sinon Déplacer le curseur.

X = XMAX	Y	N
setcurs (Y, X + 1)		X
début = ?	X	
scroller vers la gauche le document dans la fenêtre	X	

Pseudo-code

```
begin
        receive (ECRAN, RENS_FEN, fichier, CURSOR);
        if ( X < RENS_FEN.coordx2 )
                begin
                        ++XCURSOR;
                        setcurs( YCURSOR,XCURSOR);
                end
        else
                begin
                        afficher ECRAN en décalant
                        de 1 caractère vers la droite;
                end
end
```

Module: Depl-mot-gau-fct-2

Entrée.

Document édité
CURSOR
RENS-FEN

Sortie.

ECRAN
document édité
CURSOR

Dynamique.

Cette fonction est déclenchée soit par le message Choix-msg-3-c soit par le message Choix-msg-3-m soit par le message Choix-msg-3-l.

Description.

Objectif:
Cette fonction permet de déplacer le curseur jusqu'au mot suivant à gauche.

Règles:
Il suffit d'effectuer le même traitement que pour le Dépl-car-gau-fct-2 jusqu'à ce que l'on rencontre un blanc,un CR,un LF,un EOL,un EOP... (tout ce qui est différent d'une lettre).

Précondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND (RENS_FEN.cordy1l < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2)

Postcondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND (RENS_FEN.cordy1l < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2) AND (NBCAR=NBCAR+longueur du mot)

Traitement.

X = XMIN	Y	Y	Y	Y	N	N	N	N
carlu = FF ou CR	Y	Y	N	N	Y	Y	N	N
carlu = blanc	Y	N	Y	N	Y	N	Y	N
decre. X							X	X
dépl-l-bas ( )		X	X	X		X		
lire car. précédant							X	X
setcurs (X, Y)							X	

Pseudo-code.

```
begin
        receive (CURSOR,fichier,RENS_FEN,ECRAN);
        lire le caractère sous le curseur;
        while (car lu différent blanc ou FF ou CR ou EOL)
                et (X>RENS_FEN.cordxl+1)
                        lire le caractère précédent;
                        --X;
        while (car lu = blanc ou FF ou EOL ou CR)
                et (X>RENS_FEN.cordxl+1)
                       lire le caractère précédent;
                       --X;
        if (X > RENS_FEN.cordxl+1)
                begin
                   setcurs(Y,X);
                   return;
                end
        déplacement-l-haut ( );
        return;
end
```

Module: Depl-mot-dr-fct-2

Entrée.

Document edité
CURSOR
RENS-FEN

Sortie.

CURSOR
Document edité

Dynamique.

Cette fonction est déclenchée soit par le message Choix-msg-2-c, par le message Choix-msg-2-m ou soit par le message Choix-msg-2-l.

Description.

Objectif:
Cette fonction permet de déplacer le curseur d'un mot vers la droite.

Règles:
Même traitement que dans le cas du Dépl-mot-gau-fct.

Précondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND (RENS_FEN.cordy1l < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2)

Postcondition.

(il existe RENS_FEN courante) AND (il existe document courant) AND (RENS_FEN.cordy1l < Y < RENS_FEN.cordy2) AND (RENS_FEN.cordx1 < X < RENS_FEN.cordx2) AND (NBCAR=NBCAR+longueur du mot)

Traitement.

X = XMAX	Y	Y	Y	Y	N	N	N	N
carlu = FF ou CR	Y	Y	N	N	Y	Y	N	N
carlu = blanc	Y	N	Y	N	Y	N	Y	N
increm. X							X	X
dépl-l-bas ( )		X	X	X		X		
lire car. suivant							X	X
setcurs (X, Y)							X	

Pseudo-code.

```
begin
   receive (fichier, RENS-FEN,CURSOR,ECRAN) from gestion
   des primitives;
   lire le car. sous le curseur;
   while (car.lu différent blanc)(car.lu n'est pas = FF
          ou CR ou EOL)
                        (X < RENS_FEN.cordx2-1)
                        {lire le car. suivant;
                        ++ X;
                        };
   while (car.lu = blanc)(X < RENS_FEN.cordx2-1)(car.lu
                n'est pas = FF ou CR ou EOL)
                        lire le caractère suivant;
                        ++X;
                        };
   if (X < RENS_FEN.cordx2-1)(car.lu différent FF
      ou CR ou EOL)
                        {
                        setcurs (Y, X)
                        return;
                        }
   dépl-l-bas ( );
end
```

Module: Retour-panique.

Entrée.

Table des descripteurs des documents,
Documents.

Sortie.

Documents édités

Dynamique.

Cette phase est déclenchée à la fin de la phase Initialisation si le contenu du message Choix-msg-0 est R ou Retour-panique

Description.

Ojectifs :
Cette phase est une phase permettant de revenir dans l'édition de documents dans la situation exacte dans laquelle on l'avait quitté auparavant par la fonction Panique-fct-2.

Règles :
Elle affiche la liste des documents manipulés dans la précédente session et déclenche la phase Edition-ph-2.

Précondition.

(choix-util-0 = R ou r) AND (il existe une edit-commande)

Postcondition.

(choix-util-0 = R ou r) AND (il existe une edit-commande) AND ((il existe un fichier PANIQUE AND il existe les fichiers temporaires AND déclenchement de l'édition) OR (il existe un fichier PANIQUE AND (il n'existe pas de fichier temporaire) OR (il n'existe pas de fichier PANIQUE))

Traitement.

Phase permettant de revenir dans l'édition de documents dans le cours d'une session qui a été interrompue par la fonction panique-ph-3.

fichier panique (il existe)	Y		N
fichiers temporaires existent	Y	N	-
ouverture des fichiers temporaires	X		
restaurer la table des Desc.	X		
restaurer RENS_FEN	X		
détruire fichier panique	X		
erreur		9	10

Pseudo-code.

```
begin
        initialisation : i = 0;
        if fichier panique n'existe pas then
                                    begin
                                       erreur(10);
                                       return;
                                    end
        while (read (fichier-panique) différent EOF)
                begin
                        DESCRTAB [i] = enreg;
                        if DESCRTAB [i].a = 1 then
                                    begin
                                       activation du document;
                                    end
                end
        call édition;
end
```

Module: Formatage-fct

Entrée.

Document édité
Format

Sortie.

Document édité
Format

Dynamique.

Déclenchée lors de la création d'un document ou lors de la fonction Format.

Description.

Objectif:
Cette fonction effectue la vérification des valeurs du format et permets à l'utilisateur de rentrer des nouvelles valeurs s'il le désire.

Règles:
Acquisition de la nouvelle valeur si nécessaire et vérification des valeurs entrées selon les règles suivantes :

- 0<= marge gauche <= 99 caract.
- 0<= largeur de texte <= 99 caract.
- 1<= interligne <=4.
- 0<= espace haut de page < 66 caract.
- 0<= longueur de texte sur la page <=66 lignes.
- 0<= tabulateur <= 66 lignes.
- fonte ∈ {messenger,italique,grec,helvesan}.
- justification à gauche ∈ {oui,non}.
- justif. à gauche et à droite ∈ {oui,non}.
- coupure des mots ∈ {oui,non}.
- vérification des coupures ∈ {non,oui}.
- 0< nombre maximum de blancs entre chaque mots <= 10.

Précondition.

(choix-util-0 = E ou e) AND (choix-util-3 = F)

Postcondition.

(choix-util-0 = E ou e) AND (choix-util-3 = F) (nom_doc appartient à {documents}) AND (fonte appartient à {messenger, helvesan, grec, italique}) AND (Vérif., coupe, justifg, justifgd appartient à {OUI, NON}) AND (marge appartient à [0,100]) AND (bas de page appartient à [0;66]) AND (interligne appartient à [1,2,3,4]) AND (marge hauteur

appartient à [0,66]) AND (tab appartient à [0,100]) AND (nombre de blanc entre deux mots appartient à [0,10])

Traitement.

On caractérise le format que le document aura à sa sortie sur l'écran et sur l'imprimante en changeant à son gré les valeurs prises par défaut. On peut définir le format d'un fichier qu'on a pas encore créé.

0 < MARGE < 100	Y	N
erreur		N
recommencer		X
valeur suivante	X	

0 < largeur-TXT < 100	Y	N
erreur		X
recommencer		X
valeur suivante	X	

0 < interligne < 5	Y	N
erreur		X
recommencer		X
valeur suivante	X	

0 < marge-haut < 66	Y	N
erreur		X
recommencer		X
valeur suivante	X	

0 < fin-de-page < 66	Y	N
erreur		X
recommencer		X
valeur suivante	X	

0 < tabulateur[i] < 100	Y	N
erreur		X
recommencer		
valeur suivante	X	

fonte appartient à {messenger, grec, helvesan,italique}	Y	N
erreur	Y	N
recommencer		X
valeur suivante	X	

justifie à gauche = oui,non	Y	N
recommencer		X
erreur		X
valeur suivante	X	

justifie à gauche, à droite appartient à {OUI,NON}	Y	N
recommencer		X
erreur		X
valeur suivante	X	

vérif. appartient à {OUI,NON}	Y	N
recommencer		X
erreur		X
valeur suivante	X	

nombre max.appartient à {1,2,3,4,.}	Y	N
recommencer		X
erreur		X
fin	X	

Pseudo-code.

```
begin
            receive (fichier) from format OR Crèation OR
            Modification;
début :     afficher valeur FORMAT;
marge :     positionner le curseur sur 1ère valeur;
            get cla;
            if get cla = CR then go to finval;
            if get cla = tab then go to largeur;
            modifier marge;
test 1 :    if marge < 0 or marge > 210 then erreur (7);
                                                modifier marge;
                                                go to test 1;
largeur :   positionner le curseur sur 2ème valeur;
            get cla;
            if get cla = CR then go to fin val;
            if get cla = tab then go to inter;
            modifier largeur-txt;
test 2 :    if largeur-txt < 0 or > 210
                        then begin
                                erreur(7);
                                modifier largeur-txt;
                                go to test-2;
                             end
inter :     positionner le curseur sur 3ème valeur;
            get cla;
            if get cla = CR then go to fin-val;
            if get cla = TAB then go to marge-ht;
            if get cla = TAB then go to marge-ht;
            modifier interligne;
test 3 :    if interligne<0 ou interligne >5
                        then begin
                                erreur(7);
                                modifier interligne;
                                go to test 3;
                             end
marge-ht :  positionner le curseur sur la quatrième valeur;
            get cla;
            if get cla = CR then go to fin-val;
            if get cla = TAB then go to fin-page;
            modifier marge haut;
test 4 :    if marge-hauteur < 0 or > 586
                        then begin
                                erreur (7);
                                modifier marge hauteur;
                                go to test 4;
                             end
finpage :  positionner le curseur sur la cinquième valeur;
           get cla;
           if get cla = "CR" then go to fin-val;
           if get cla = TAB then go to tabulat;
           modifier fin-page;
test 5 :  if fin-page < 0 or > 160 then
                          begin
                              erreur(7);
                              modifier fin-page;
```

```
                                go to finpage;
                              end
tabulat : positionner le curseur sur la 6 ième valeur;
          b-tab : i = i + 1;
                        setcurs (4,X + i);
                        get cla;
                        if get cla = CR then go to fin-val;
                        if get cla = Tab then go to fonte;
                        if get cla = "space" then go to
                        b-tab;
                        modifier tab[i];
test 6 : if tab[i] = 0 or > 400 then
                                  begin
                                       erreur(7);
                                       modifier tab (i);
                                       go to test 6;
                                  end

font : positionner le curseur sur la 7 ième valeur;
       get cla;
       if get cla = CR then go to fin-val;
       if get cla = tab then go to justgan;
       modifier fonte;
test 7 : if fonte différent messenger, grec, helvesan,
               italique
               then
                        begin
                             erreur(7);
                             modifier fonte;
                             go to test 7;
                        end
justgau : positionner le curseur sur la 8ième valeur;
          get cla;
          if get cla = CR then go to fin-val;
          if get cla = tab then go to justgd;
          modifier justgau;
test 8 : if justgau différent OUI OR NON OR YES OR NO
                   then begin
                                erreur(7);
                                modifier;
                                go to test 8;
                        end
justgd : positionner le curseur;
         get cla;
         if get cla = CR then go to fin-val;
         if get cla = Tab then go to vérif;
         modifier justgd;
test 9 : if justgd n'appartient pas à {OUI,NON,YES,NO}
                        then begin
                                erreur (7);
                                modifier;
                                go to test 9;
                             end
vérif : positionner le curseur;
           get cla;
           if get cla = CR then go to fin-val;
           if get cla = tab then go to nombre;
```

```
            modifier vèrif;
test 10 :    if vèrif appartient à {OUI,NON,YES,NO} then
                                    begin
                                       erreur (7);
                                       modifier;
                                       go to test 10;
                                    end
nombre : positionner le curseur;
         get cla ;
         if get cla = CR then go to fin-val;
         if get cla = TAB then go to de ´but;
         modifier nombre-max;
test 11 :if nombre-max < 0 then
                         begin
                            erreur (7);
                            modifier;
                            go to test 11;
                         end
fin-val : end;
```

Module: Delimiter_texte

Entrè.

Document èditè

Sortie.

Document èditè
Numèro de ligne debut,numèro de ligne fin
Numèro de caractère dèbut, numèro de caractère fin

Dynamique.

Cette fonction est dèclenchèe par les fonctions suivantes :

- copie-fct
- transfert-fct
- efface-fct
- caractère-fct

Description.

Objectif:
Cette fonction a pour but de dèlimiter une portion de texte sur lequel on va effectuer une modification.

Prècondition.

(il existe un document courant) AND (il existe un RENS-FEN courant) AND [((choix-util-2 = Crèation) AND ((choix-util-2-c = Copie) OR (choix-util-2-c = Transfert) OR (choix-util-2-c = Efface) OR (choix-util-2-c = caractère))) OR ((choix-util-2 = Modification) AND ((choix-util-2 = Copie) OR (choix-util-2-m = Transfert) OR (choix-util-2-m = Efface) OR (choix-util-2-m = caractère)))]

Postcondition.

(il existe un document courant) AND (il existe un RENS-FEN courant) AND [((choix-util-2 = Crèation) AND ((choix-util-2-c = Copie) OR (choix-util-2-c = Transfert) OR (choix-util-2-c = Efface) OR (choix-util-2-c = caractère))) OR ((choix-util-2 = Modification) AND ((choix-util-2 = Copie) OR (choix-util-2-m = Transfert) OR (choix-util-2-m = Efface) OR (choix-util-2-m = caractère)))] AND (numèro leg deb, numèro leg fin, numèro car deb, numèro car fin dèfinis)

Traitement.

Si ce qui a ètè lu au clavier est un dèplacement de curseur
Sinon effectuer le dèplacement du curseur.

Si c'est un CR mémoriser le numéro de ligne début.
Ensuite, faire de même pour mémoriser le numéro de ligne de la fin de la portion de texte.

Pseudo-code

```
begin
        receive (fichier,RENS_FEN,CURSOR);
        while (getcla != 0 do
                begin
                        if (CTRL_X) Dépl-4-bas;
                        if (CTRL_C) Dépl-1-bas;
                        if (CTRL_V) Dépl-fin-txt;
                        if (CTRL_E) Dépl-4-haut;
                        if (CTRL_R) Dépl-1-haut;
                        if (CTRL_T) Dépl-déb-txt;
                        if (CTRL_F) Dépl-car-gau;
                        if (CTRL_G) Dépl-mot-gau;
                        if (CTRL_D) Dépl-car-dr;
                        if (CTRL_S) Dépl-mot-dr;
                end
        cadre.coordy1 = no_lig_cour;
        cadre.coordx1 = CURSOR.x - RENS_FEN.cordx1;
        while (getcla != 0 do
                begin
                        if (CTRL_X) Dépl-4-bas;
                        if (CTRL_C) Dépl-1-bas;
                        if (CTRL_V) Dépl-fin-txt;
                        if (CTRL_E) Dépl-4-haut;
                        if (CTRL_R) Dépl-1-haut;
                        if (CTRL_T) Dépl-déb-txt;
                        if (CTRL_F) Dépl-car-gau;
                        if (CTRL_G) Dépl-mot-gau;
                        if (CTRL_D) Dépl-car-dr;
                        if (CTRL_S) Dépl-mot-dr;
                end
        cadre.coordy2 = no_lig_cour;
        cadre.coordx2 = CURSOR.x - RENS_FEN.cordx1;
        if (cadre.coordy2 < cadre.coordy1)
                begin
                        inverser x2 et x1;
                        inverser y2 et y1;
                end
end
```

Module: Gestion des primitives

Entrée.

Document_édité
RENS-FEN

Sortie.

Document-édité
RENs-FEN

Dynamique.

La gestion des primitves est appelée lors de la création et de la modification.

Description.

règles:

Choix-util.	JUS	PAG	VIS	SP	MTR	COP	REC	CHA	MIN	MAJ	TRS	#
Justificat.	X											
Pagination		X										
Vis.car.sp.			X									
Saut page				X								
Montre					X							
Copie						X						
Recherche							X					
Change								X				
Minuscule									X			
Majuscule										X		
Transfert											X	
Autre choix												X

Autre choix	CAR	DEF	EFF	CEN	INS	DEL	BS	l_H	l_B	4_H	4_B	#
Caractere	X											
Defaire		X										
Efface			X									
Centre				X								
Insertion					X							
Delete						X						
Backspace							X					
Depl_l_haut								X				
Depl_l_bas									X			
Depl_4_haut										X		
Depl_4_bas											X	
Autre choix												X

Autre choix	l_G	l_D	M_G	M_D	D_T	F_T	RET
Depl_car_g	X						
Depl_car_d		X					
Depl_mot_g			X				
Depl_mot_d				X			
Depl_deb_tx					X		
Depl_fin_tx						X	
Retour							

Précondition.

(il existe RENS-FEN courant) AND (il existe document-courant) AND ((choix-util-2 = Création) OR (choix-util-2 = Modification))

Postcondition.

(il existe RENS-FEN courant) AND (il existe document-courant) AND [((choix-util-2 = Création) AND (choix-util-2-c appartient a {deplacements})) OR ((choix-util-2 = Modification) AND (choix-util-2-m appartient a {deplacements}))]

Traitement.

lecture du choix de la primitive à réaliser et selon ce choix appel à la routine adéquate.

Pseudo-code.

```
begin
        receive (fichier,RENS-FEN) from création or
        from modification;
        initialisation : RET = OFF;
                         sw_ins=OFF;
        while (RET = OFF) do
        begin
              lire-choix (clavier);
              if (choix(clavier) = MENU)
                then
                   begin
                     positionner au début du menu;
                     if (sw-ins=ON) recopier l'écran
                                    dans le fichier;
                     traitement-primitive (fichier);
                   end
                else
                   begin
                        if CTRL-R
                           begin
                             if (sw-ins=ON) then
                                recopier l'écran
```

```
            dans le fichier;
               sw-ins=off;
          call depl-1-haut;
       end
    if CTRL-E
       begin
         if (sw-ins=ON) then
                    recopier l'écran
                    dans le fichier;
                    sw-ins=off;
           call depl-4-haut;
         end
    if CTRL-C
         begin
           if (sw-ins=ON) then
                  recopier l'écran
                  dans le fichier;
                  sw-ins=off;
           call depl-1-bas;
         end
    if CTRL-X
         begin
           if (sw-ins=ON) then
                  recopier l'écran
                  dans le fichier;
                  sw-ins=off;
           call depl-4-bas;
         end
    if CTRL-T
         begin
           if (sw-ins=ON) then
                  recopier l'écran
                  dans le fichier;
                  sw-ins=off;
           call depl-deb-txt;
         end
    if CTRL-V
         begin
           if (sw-ins=ON) then
                  recopier l'écran
                  dans le fichier;
                  sw-ins=off;
           call depl-fin-txt;
         end
    if CTRL-F
         begin
           if (sw-ins=ON) then
                  recopier l'écran
                  dans le fichier;
                  sw-ins=off;
           call depl-car-dr;
         end
    if CTRL-D
         begin
           if (sw-ins=ON) then
                  recopier l'écran
                  dans le fichier;
```

```
                                        sw-ins=off;
                                    call depl-car-gau;
                                end
                        if CTRL-G
                                begin
                                    if (sw-ins=ON) then
                                            recopier l'écran
                                            dans le fichier;
                                            sw-ins=off;
                                    call depl-mot-dr;
                                end
                        if CTRL-S
                                begin
                                    if (sw-ins=ON) then
                                            recopier l'écran
                                            dans le fichier;
                                            sw-ins=off;
                                    call depl-mot-gau;
                                end
                        if BS call backspace ( );
                        if DEL
                                begin
                                            call del( );
                                            sw_ins = ON;
                                end
                        if "n'est pas égal"
                                begin
                                    call insertion(choix);
                                    sw-ins=ON;
                                end
                    end
            end
    end
```

Pseudo-code : traitement-primitive (fichier).

```
begin
        receive (fichier) from gestion des primitives;
        initialisation RET = OFF;
        lect-ch-primitive;
        while choix n'est pas égal à retour do
           begin
             if choix = justifie then call justifie.
             if choix = pagine then call pagine;
             if choix = visual then call visualisation;
             if choix = montre then call montre;
             if choix = cherche then call cherche;
             if choix = change then call change;
             if choix = minuscule then call minuscule;
             if choix = majuscule then call majuscule;
             if choix = transfert then call transfert;
             if choix = caractère then call caractère;
             if choix = défaire then call défaire;
             if choix = encadre then call encadre;
             if choix = efface then call efface;
             if choix = saut de page then call saut de page;
             if choix = centre then
                                        begin
                                        call centre;
                                        sw_ins = ON;
                                        end
             if choix = Menu then  return = OFF
                                     go to fin;
           end
        ret = ON;
fin : send (ret);
```

Module: take-coord

Entrée.

CURSOR

Sortie.

CURSOR
RENS-FEN

Dynamique.

Cette fonction est déclenchée par le module de CREATION et le module de MODIFICATION.

Description.

Objectif:
Cette fonction permet à l'utilisateur de déterminer les coordonnées de la fenêtre dans laquelle il désire travailler.

Précondition.

Rien

Postcondition.

Rien

Traitement.

Lire la première coordonnée X1,Y1.
Lire la seconde coordonnée Y2,X2.
Verifier que 0<Xi<=63 et 4<Y<=24.
Si X2<X1 intervertir X2 et X1.
Si Y2<Y1 intervertir Y2 et Y1.
Si (X2-X1)<16 recommencer.
Si (Y2-Y1)<36 recommencer.

Pseudo-code.

```
begin
       receive (CURSOR,RENS-FEN);
début: while (getchar( ) différent CR) do
                begin
                  if CTRL_C
                        begin
                           Y=Y+1;
                           setcurs(Y,X);
                           goto endw;
                        end
                  if CTRL_X
                        begin
```

```
                    Y=Y+4;
                    setcurs(Y,X);
                    goto endw;
                end
            if CTRL_R
                begin
                    Y=Y-1;
                    setcurs(Y,X);
                    goto endw;
                end
            if CTRL_E
                begin
                    Y=Y-4;
                    setcurs(Y,X);
                    goto endw;
                end
        if CTRL_F
                begin
                    X=X+1;
                    setcurs(Y,X);
                    goto endw;
                end
        if CTRL_D
                begin
                    X=X-1;
                    setcurs(Y,X);
                    goto endw;
                end
        if CTRL_G
                begin
                    X=X+4;
                    setcurs(Y,X);
                    goto endw;
                end
        if CTRL_S
                begin
                    X=X-4;
                    setcurs(Y,X);
                    goto endw;
                end
        if CTRL_T
                begin
                    Y=2;
                    setcurs(Y,X);
                    goto endw;
                end
        if CTRL_V
                begin
                    Y=23;
                    setcurs(Y,X);
                    goto endw;
                end
end
RENS_FEN.cordx1=X;
RENS_FEN.cordy1=Y;
while (getchar( ) different CR) do
        begin
```

```
if CTRL_C
      begin
         Y=Y+1;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_X
      begin
         Y=Y+4;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_R
      begin
         Y=Y-1;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_E
      begin
         Y=Y-4;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_F
      begin
         X=X+1;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_D
      begin
         X=X-1;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_G
      begin
         X=X+4;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_S
      begin
         X=X-4;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_T
      begin
         Y=2;
         setcurs(Y,X);
         goto endw;
      end
if CTRL_V
      begin
         Y=23;
```

```
                        setcurs(Y,X);
                        goto endw;
                    end
            end
            RENS_FEN.cordx2=X;
            RENS_FEN.cordy2=Y;
            if (RENS_FEN.cordx2<RENS_FEN.cordx1) inverser;
            if (RENS_FEN.cordy2<RENS_FEN.cordy1) inverser;
            if (RENS_FEN.cordx2-RENS_FEN.cordx1)<36)
                                begin
                                   erreur(8);
                                   goto début;
                                end
            if (RENS_FEN.cordy2-RENS_FEN.cordy1)<16)
                                begin
                                   erreur(8);
                                   goto début;
                                end
end
```

Module de recopiage du buffer ECRAN sur le fichier.

Entrée :

ECRAN
Document édité
RENS_FEN
Numéro de ligne courante
CURSOR

Sortie :

Document édité

Dynamique.

Ce module est déclenché lorsque il y a eu de l'insertion et que l'utilisateur veut effectuer une autre primitive différente de l'insertion.

Description.

Ce module effectue le sauvetage de l'environnement courant du document sur le fichier temporaire.

Précondition.

Rien

Postcondition.

Rien

Traitement.

Il suffit de boucler sur l'indice de ligne et recopier toutes les lignes du buffer ECRAN qui sont remplies.

Pseudo-code.

```
begin
    receive (no_ligne,fichier);
    sauter l'ENTETE et le FORMAT;
    sauter les lignes précédant no-ligne-deb;
    while (i<24)and(ECRAN[i][0] != ZERO)
       begin
       write_rand(fichier,no_ligne+i+1,ECRAN[i],BUFSIZE);
       ++i;
       end
end
```

Module d'effacage total de l'écran.

Efface l'écran à partir de la ligne 0 jusqu'à la ligne 23 (c-à-dire 24 lignes) sur une longueur de 80 caractères (0 à 79). Ce module est réalisé par une fonction du VT-100 lors du travail sur le PDP-11.

Module de crêation d'une fenêtre.

Entrêe.

RENS-FEN

Sortie.

RENS-FEN
Fenêtre-document

Prêcondition.

Rien

Postcondition.

Rien

Dynamique.

Cette fonction est appelée dans le module de crêation afin de dessiner la fenêtre de visualisation du document. Elle est aussi appelée dans le module de modification,le module format, le module de lecture lorsque le document n'est pas encore dans la table des descripteurs.

Description.

Ce module trace les lignes qui dêlimite la fenêtre dont les coordonnées ont êtê mêmorisêes par take-coord.

Traitement.

Il suffit de passer en graphique et de dessiner les lignes adêquates entre les bornes RENS_FEN.

Module d'effacage du contenu de la fenêtre.

Entrée.

Rien

Sortie.

Rien

Description.

Efface le contenu d'une fenêtre grâce aux coordonnées de cette fenêtre qui ont eété mémorisées. Il suffit d'écrire des blancs entre les bornes de la fenêtre.

Pseudo-code.

```
begin
        receive (RENS_FEN);
        i=0;
        while (i<=RENS_FEN.lon_fen-7)
           begin
              setcurs(RENS_FEN.cordyll+i+1,RENS_FEN.cordxl+1);
              for (i=0;i,RENS_FEN.larg_fen-1;++i)
                      write un blanc;
           end
end
```

Module setcurs (X,Y).

Entrée.

x,y

Sortie.

CURSOR

Précondition.

Rien

Postcondition.

Rien

Traitement.

Y < 0	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
Y < 23	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
X < 0	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
X > 79	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
X = 0						X				X				X		
X = 79							X				X				X	
Y = 0						X	X	X								
Y = 23										X	X	X				
positionner le curseur						X	X	X		X	X	X		X	X	X
impossible	X	X	X	X	X				X				X			

Pseudo-code.

```
begin
        receive X, Y;
        if X < 0        X = 0;
        if X > 79       X = 79;
        if Y < 0        Y = 0;
        if Y > 23       Y = 23;
        positionner le curseur;
end
```

Module de copiage du document original sur un fichier temporaire.

Entrée.

Nom du fichier
Fichier

Sortie

Fichier temporaire
Tables des descripteurs.

Dynamique.

Ce module est déclenché dans la création et dans la lecture, la modification, le format,l'activation si le fichier n'a pas encore été édité dans la session.

Précondition.

(il existe le fichier original) AND (DESCRTAB n'est pas pleine)

Postcondition.

(il existe le fichier original) AND (DESCRTAB n'est pas pleine) AND ((il existe un fichier temporaire) OR (Message d'erreur))

Traitement.

ayant reçu le nom du fichier original, il faut composer le nom du fichier temporaire(extension du fichier sera .TEM). Une fois ce nom crée, il faut créer le fichier temporaire et recopier le fichier original sur le fichier temporaire.

Pseudo-code.

```
receive (nom-document);
begin
        composer le nom du document temporaire;
                (même préfixe mais extention différente : TEM)
        créer le fichier temporaire;
        copier le document original sur le document temporaire;
        mettre le nom du document dans la table du descripteur;
        initialiser les flags D = 0, S = 0, A = 0;
end;
```

Module de lecture du choix dans un menu.

Entrée.

CURSOR
début menu
fin menu

Sortie.

CURSOR
choix

Dynamique.

Ce module est déclenché lorsqu'il y a un choix à faire dans un menu.

Précondition.

Rien

Postcondition.

Rien

Traitement.

Il faut se déplacer dans le menu tant que l'tuilisateur ne sélectionne pas un choix. Quand le choix a été selectionné il faut l'envoyer au programme appelant.

Pseudo-code.

```
begin
  setcurs au début du menu;
  c = get choix ( );
  while (c ! = 0
      { if c = dépl-4-bas then Y = Y + 4;
                                setcurs;
        if c = dépl-4-haut then Y = Y - 4
                                setcurs;
        if c = dépl-1-bas then  Y = Y + 1;
                                setcurs;
        if c = dépl-1-haut then Y = Y - 1;
                                setcurs;
        if c = dépl-début then Y = début;
                                  setcurs;
        if c = dépl-fin then Y = fin;
                               setcurs.
        c = getch( );
       }
  read (écran,choix);
  send (choix) to edition-ph-2;
end
```

Module d'erreur.

Entrée.

NUMERO

Sortie.

Le message correspondant au numéro.

Voici la liste des messages d'erreur et les numéros correspondant.

Numéro	Message
1	CHOIX ERRONE ! VEUILLEZ RECOMMENCER
2	CE DOCUMENT N'EXISTE PAS
3	LE NOM DU DOCUMENT EST TROP LONG
4	CE DOCUMENT EXISTE DEJA
5	LA SUITE DE CARACTERES RECHERCHEE N'A PAS ETE TROUVEE
6	ORDRE IMPOSSIBLE DANS CE CAS
7	LA VALEUR ENTREE EST IMPOSSIBLE A SATISFAIRE
8	L'ENCADREMENT EST IMPOSSIBLE ! VEUILLEZ RECOMMENCER
9	AUCUN DOCUMENT DANS LE FICHIER PANIQUE N'A ETE EDITE NORMALEMENT
10	LE FICHIER DE PANIQUE N'EXISTE PAS
11	DOCUMENT NON CREE
12	ERREUR D'ECRITURE
13	ERREUR DE LECTURE ! RECOMMENCEZ LE CHOIX
14	ERREUR DANS LA CREATION DU DOCUMENT
15	ERREUR D'OUVERTURE DU DOCUMENT
16	LISTE DES DOCUMENTS EDITES DANS LA SESSION PRECEDENTE
17	IMPOSSIBLE DE DETRUIRE CE DOCUMENT CAR NON EDITE
18	TAPER < RETURN > POUR CONTINUER
19	IMPOSSIBLE DE SAUVER CE DOCUMENT CAR NON EDITE
20	COIN SUPERIEUR GAUCHE DE LA FENETRE PUIS < RETURN >
21	COIN INFERIEUR DROIT DE LA FENETRE PUIS < RETURN >
22	ERREUR DANS LES COORDONNEES DE LA FENETRE
23	LA LONGUEUR DE LA FENETRE DOIT ETRE > QUE 15 LIGNES
24	LA LARGEUR DE LA FENETRE DOIT ETRE > QUE 36 CARACTERES
25	COIN SUPERIEUR GAUCHE DE L'ENCADREMENT
26	COIN INFERIEUR DROIT DE L'ENCADREMENT
27	ERREUR DANS LES COORDONNES DE L'ENCADREMENT
28	NON IMPLEMENTE
29	NON JUSTIFIE
30	VOULEZ-VOUS SAUVER CE DOCUMENT SOUS UN AUTRE NOM ?
31	SI NON, TAPER < RETURN >, SI OUI, TAPER LE NOUVEAU NOM
32	DESIREZ-VOUS SAUVER LE DOCUMENT
33	LE DOCUMENT N'A PAS ETE PAGINE
34	LA SUITE DE CARACTERES EST TROP LONGUE
35	LA VALEUR DOIT ETRE COMPRISE ENTRE 0 ET 100
36	LA VALEUR DOIT ETRE COMPRISE ENTRE 0 ET 99

```
37      LA VALEUR DOIT ETRE COMPRISE ENTRE 0 ET 4
38      LA VALEUR DOIT ETRE COMPRISE ENTRE 0 ET 66
39      LA VALEUR DOIT ETRE COMPRISE ENTRE 0 ET 65
40      FONTE DISPONIBLE : MESSENGER, ITALIQUE, HELVESAN,
        GREC
41      REPONSE PERMISE : OUI, NON, oui, non
42      LA VALEUR DOIT ETRE COMPRISE ENTRE 0 ET 10
43      LA VALEUR DOIT ETRE PLUS GRANDE QUE LA PRECEDENTE
44      VOUS AVEZ ATTEINT LA FIN DU FICHIER
45      VOUS AVEZ ATTEINT LE DEBUT DU FICHIER
46      FENETRE TOTALEMENT DECALEE
47      DETERMINER LE DEBUT DE LA PORTION DE TEXTE
48      DETERMINER LA FIN DE LA PORTION DE TEXTE
```

Description.

Ce module effectue l'impression du message d'erreur dont le numéro a été passé comme paramètre.

3. Messages.

MESSAGE: Edit

SYNONYME: Edit-command

DYNAMIQUE: Ce message déclenche l'éditeur et en premier lieu , la phase Initialisation.

DESCRIPTION: Ce message est la commande qui exécute tout le système d'édition.

MESSAGE: Menu-initialisation

SYNONYME: Ecran-1

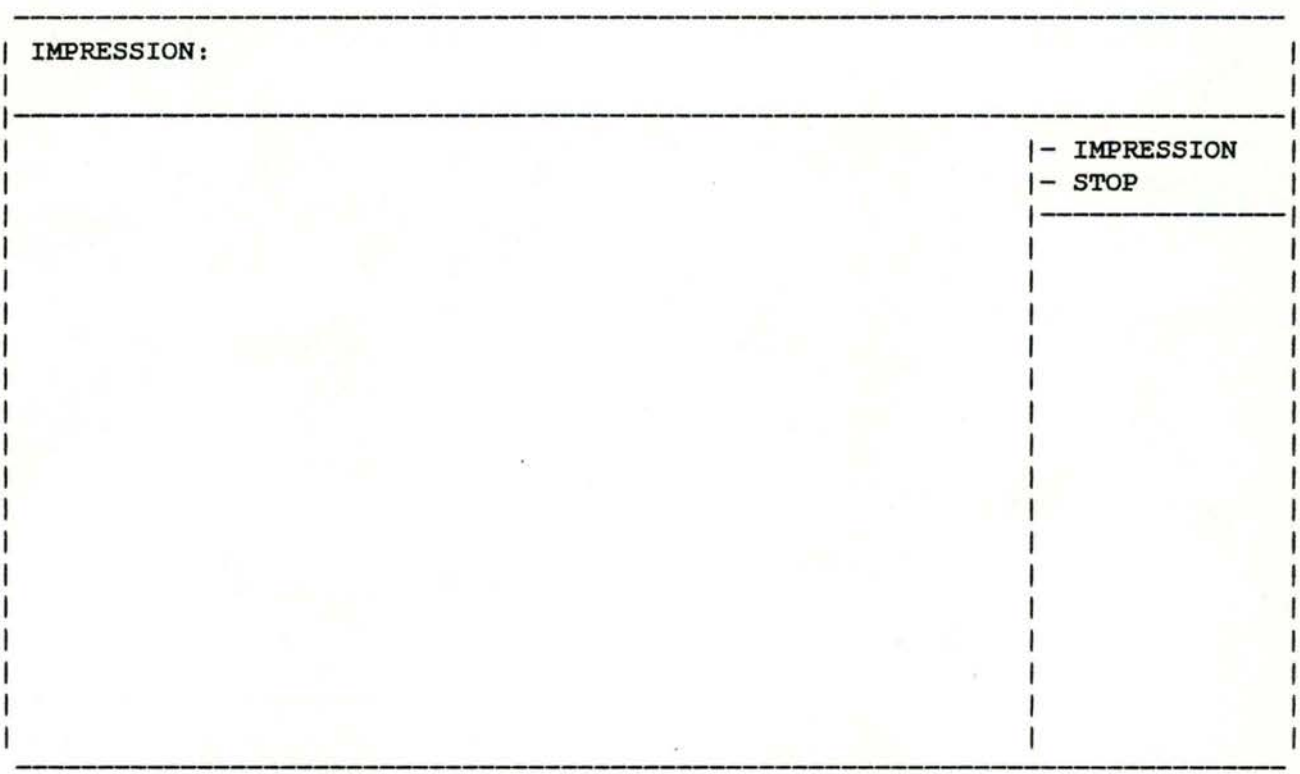
DESCRIPTION:

```
+-----------------------------------------------------------------------------+
|  EDITEUR                                                                    |
|-----------------------------------------------------------------------------|
|                                                                             |
|                                  M E N U                                    |
|                                  =======                                    |
|                                                                             |
|                                                                             |
|                          _ E(dition)                                        |
|                                                                             |
|                          _ I(mpression)                                     |
|                                                                             |
|                          _ R(etour-panique)                                 |
|                                                                             |
|                          _ S(top)                                           |
|                                                                             |
|                                                                             |
|                              VOTRE CHOIX ? : _                              |
|                                                                             |
+-----------------------------------------------------------------------------+
```

MESSAGE: Ecran-impr

SYNONYME: Ecran-2

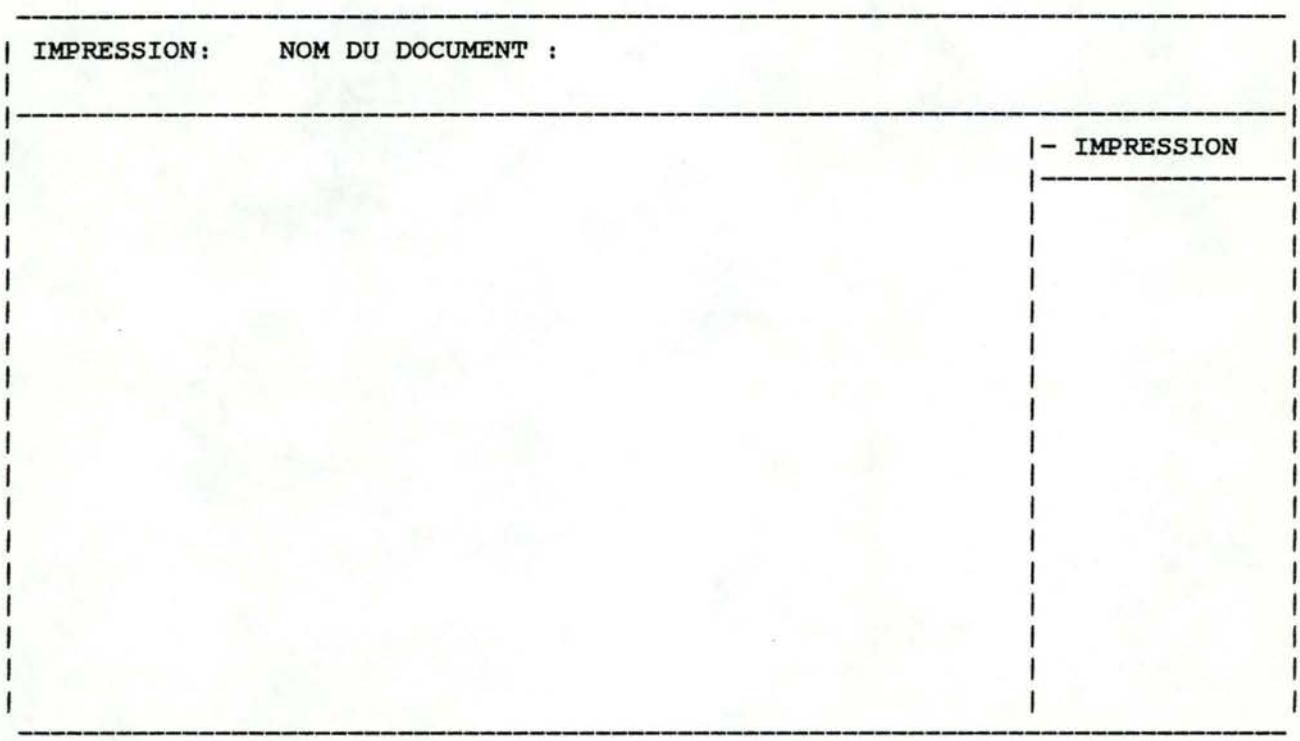
DESCRIPTION:



MESSAGE: Ecran-impression

SYNONYME: Ecran-2-i

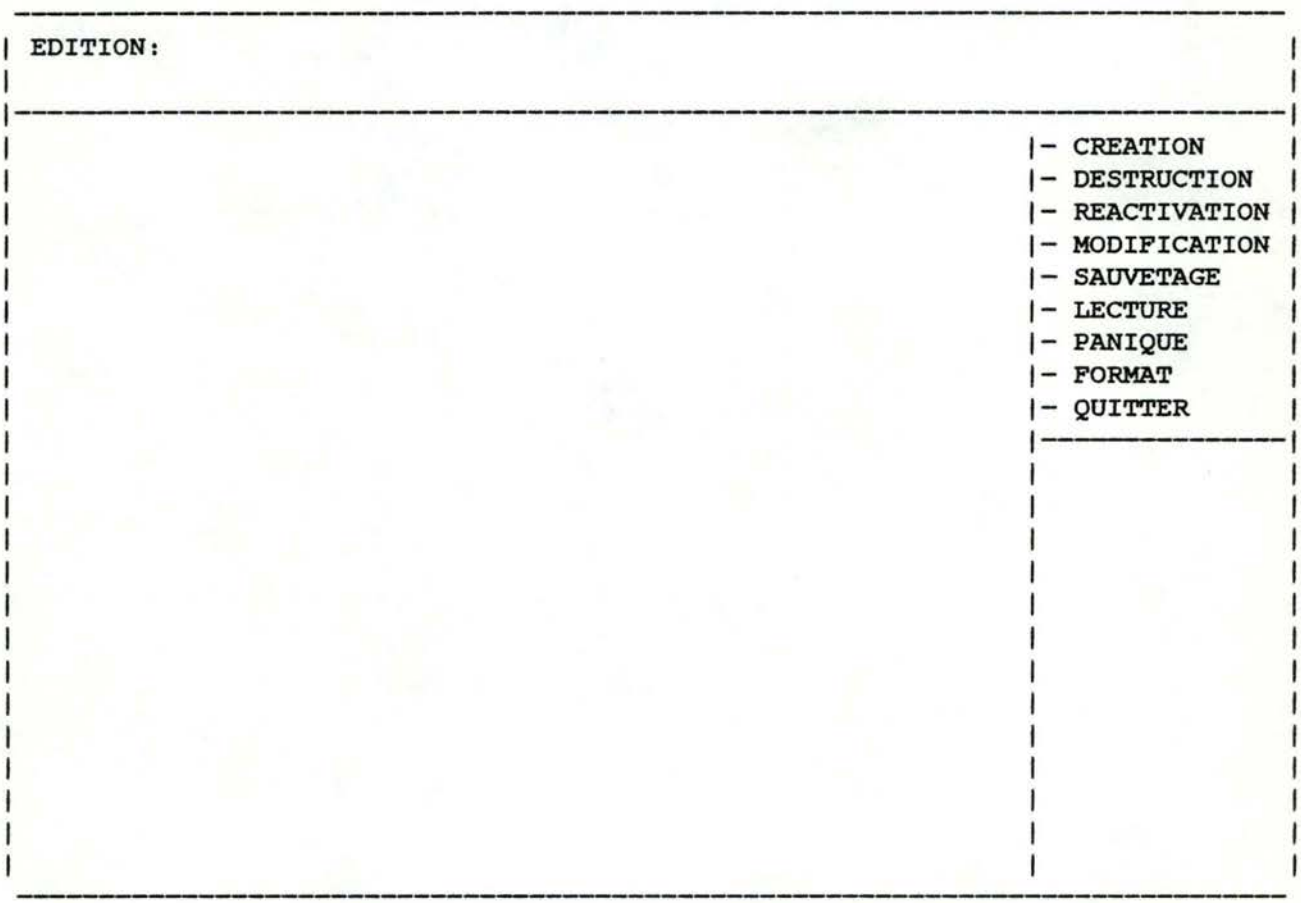
DESCRIPTION:



MESSAGE: Ecran-édit

SYNONYME: Ecran-2

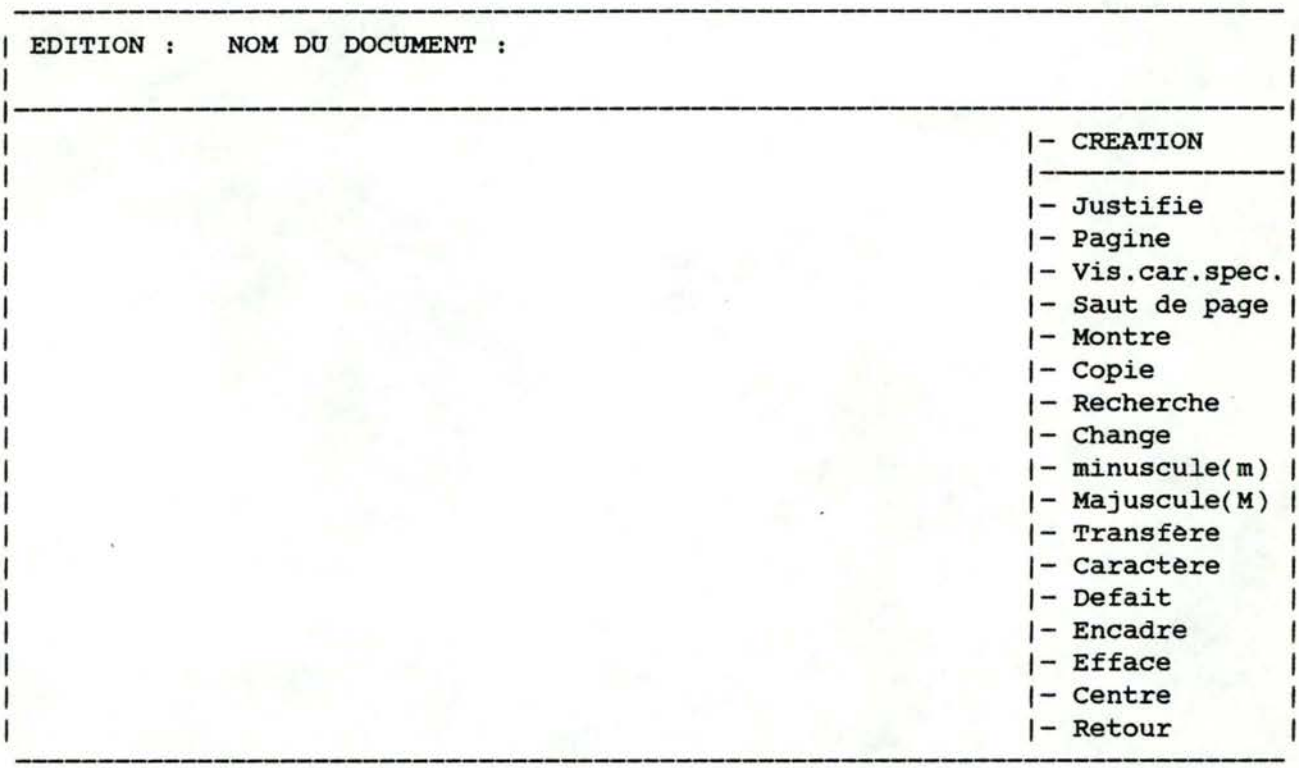
DESCRIPTION:



MESSAGE: Ecran-creat

SYNONYME: Ecran-2-c

DESCRIPTION:



MESSAGE: Ecran-dest

SYNONYME: Ecran-2-d

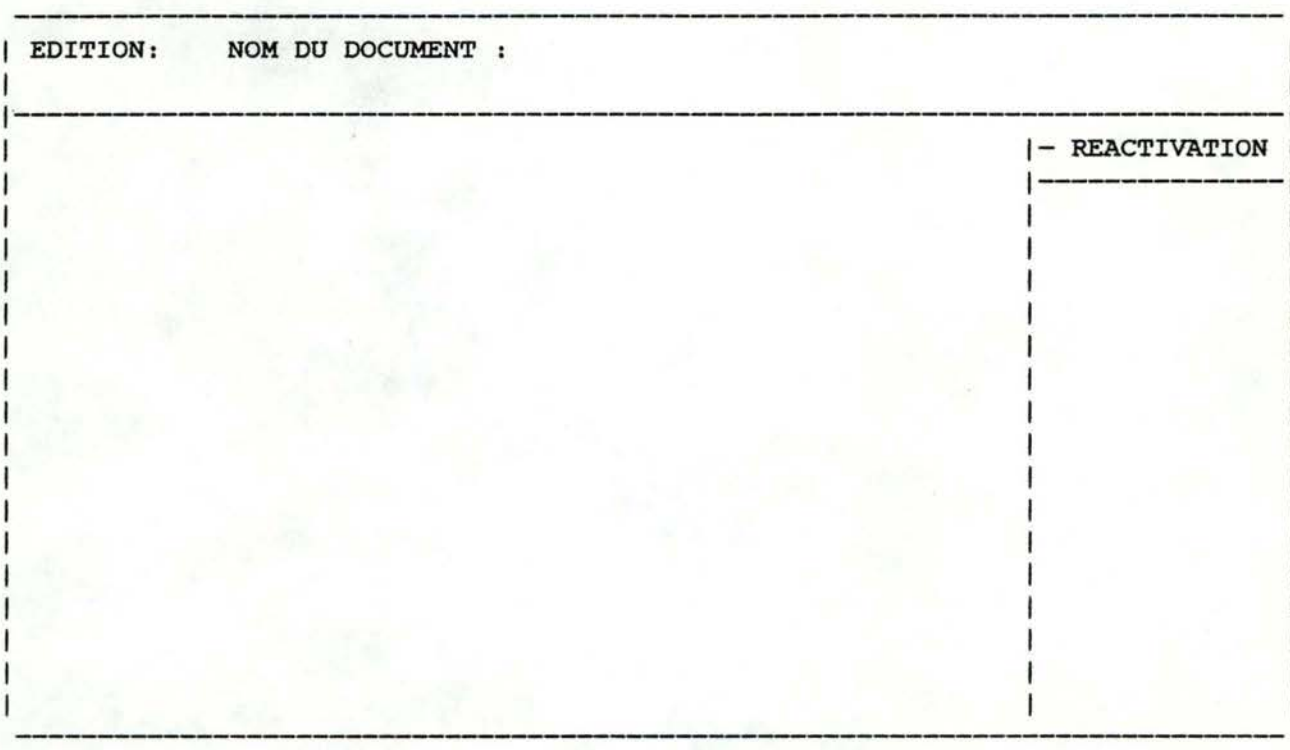
DESCRIPTION:

EDITION: NOM DU DOCUMENT :	
	- DESTRUCTION

MESSAGE: Ecran-act

SYNONYME: Ecran-2-a

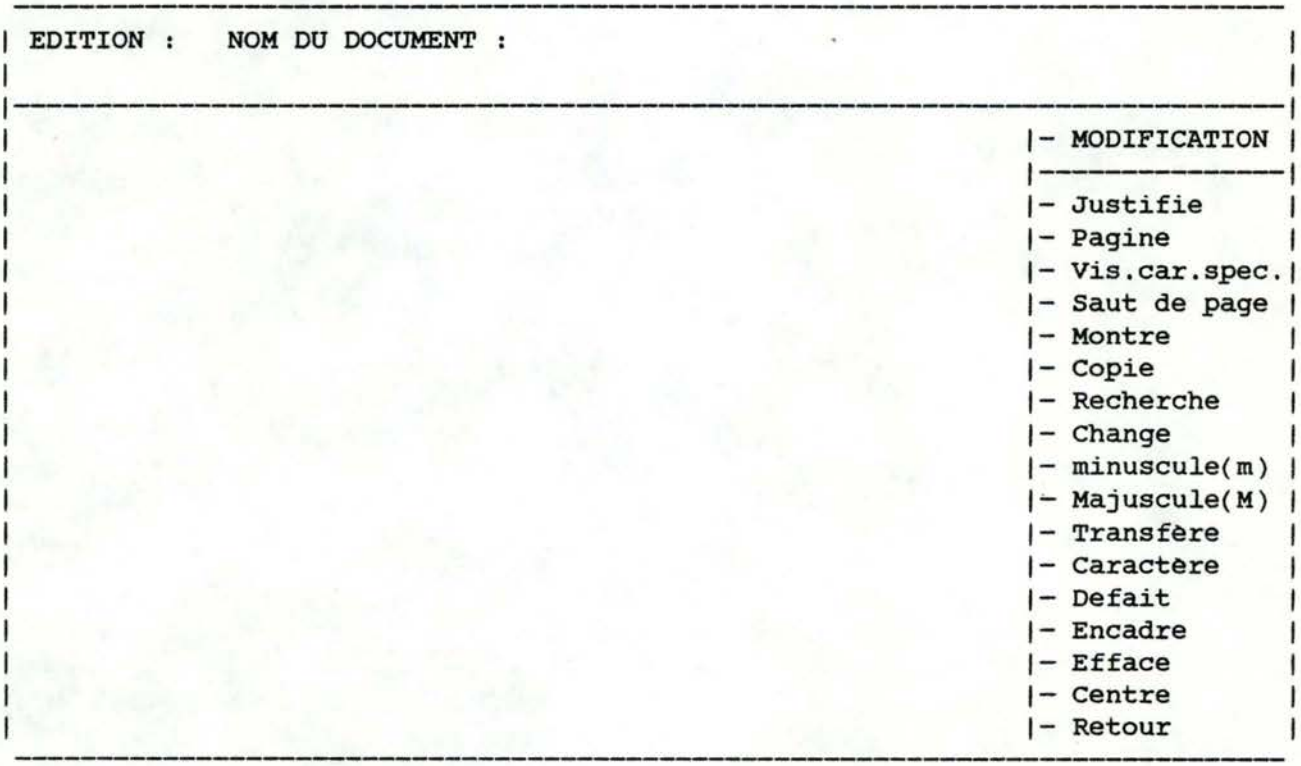
DESCRIPTION:



MESSAGE: Ecran-mod

SYNONYME: Ecran-2-m

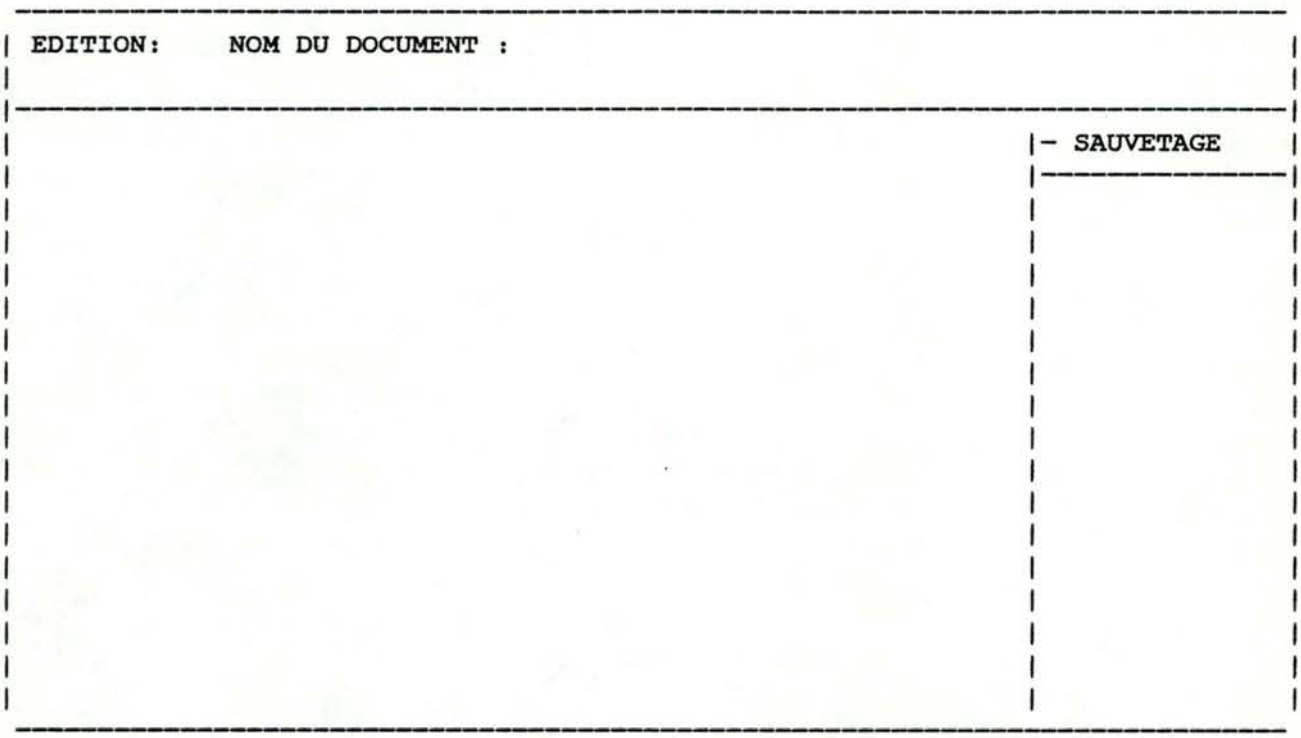
DESCRIPTION:



MESSAGE: Ecran-sauv

SYNONYME: Ecran-2-s

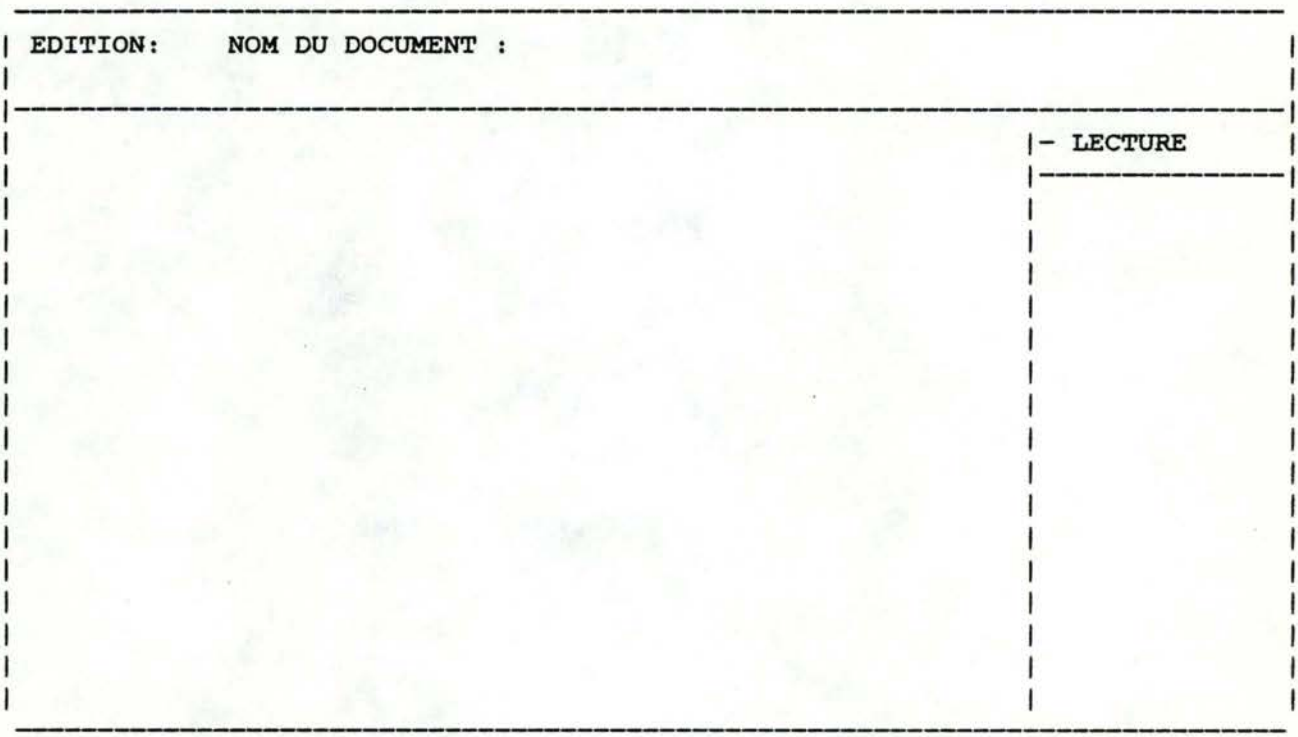
DESCRIPTION:



MESSAGE: Ecran-lect

SYNONYME: Ecran-2-1

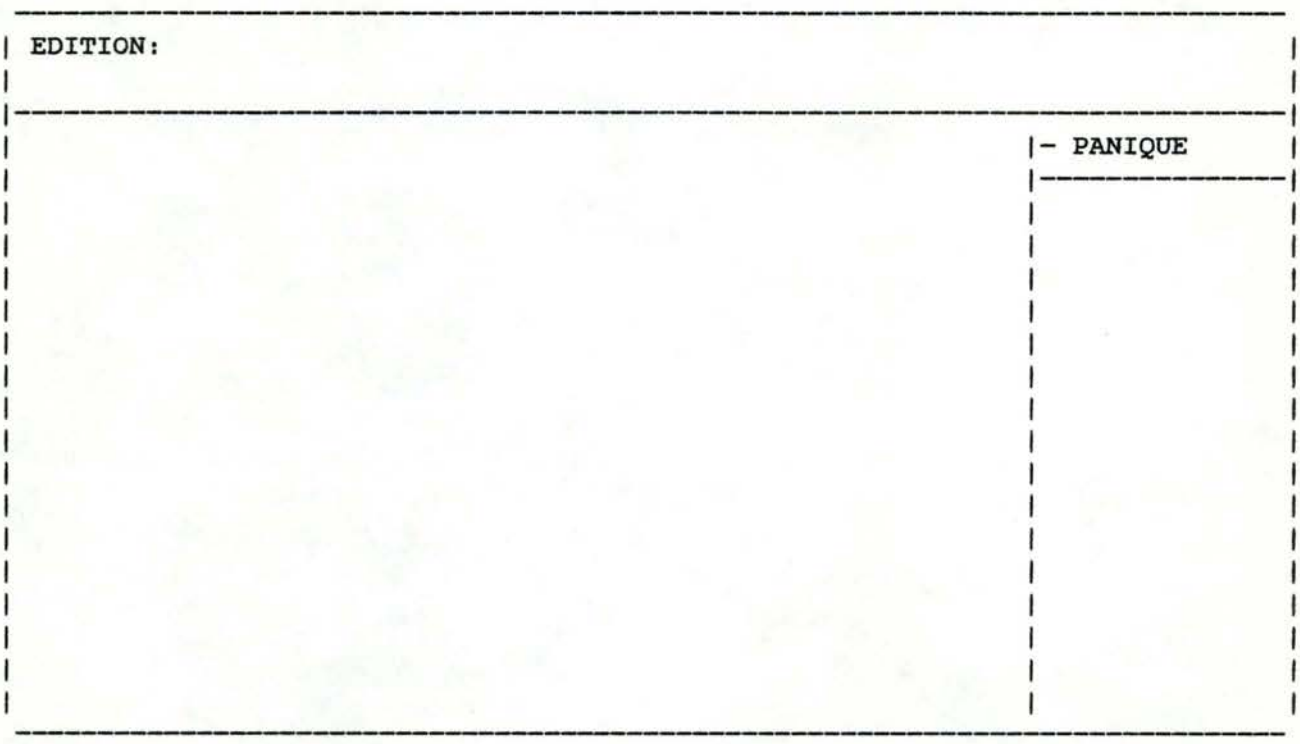
DESCRIPTION:



MESSAGE: Ecran-pan

SYNONYME: Ecran-2-p

DESCRIPTION:



MESSAGE: Ecran-fmt

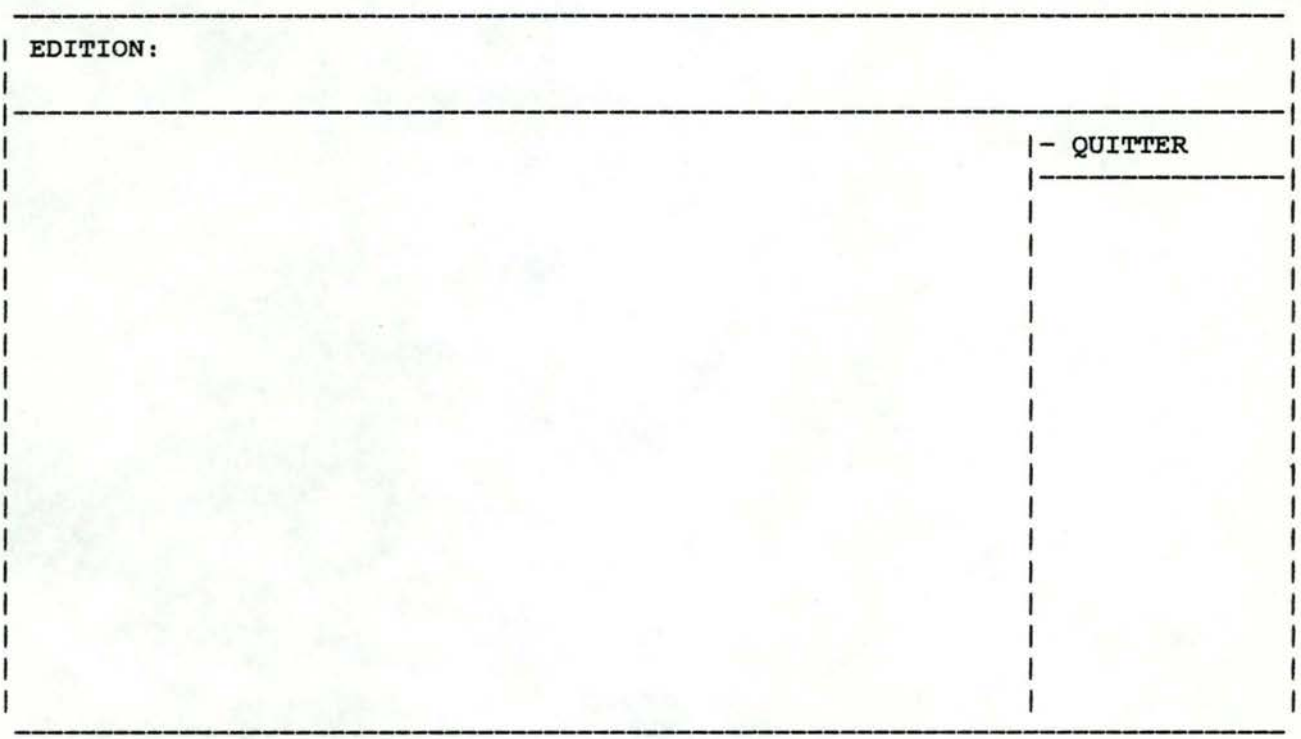
DESCRIPTION:

```
EDITION :  NOM DU DOCUMENT :
|
|-------------------------------------------------------------------------------|
|                                                                               |- FORMAT
|  ---------------------------------------------------------                    |-------------|
|  | DOCUMENT :                Date Creat. :                |                    |             |
|  |                           Date m-a-j. :                |                    |             |
|  |--------------------------------------------------------|                    |             |
|  | Marge gauche                        :  10              |                    |             |
|  | Largeur du texte                    :  80              |                    |             |
|  | Interligne                          :   1              |                    |             |
|  | Marge haut                          :   5              |                    |             |
|  | Fin de page                         :  64              |                    |             |
|  | Tabulateurs                         :   0   0   0   0   0                   |             |
|  | Fonte                               : MESSENGER        |                    |             |
|  | Justification à gauche              : OUI              |                    |             |
|  | Justification à gauche et a droite: NON                |                    |             |
|  | Coupure des mots                    : NON              |                    |             |
|  | Verification des coupures           : NON              |                    |             |
|  | Nombre maximum blancs entre 2 mots:   2                |                    |             |
|  ---------------------------------------------------------                    |             |
|                                                                               |             |
|                                                                               |             |
|                                                                               |             |
|                                                                               |             |
-------------------------------------------------------------------------------
```

MESSAGE: Ecran-quit

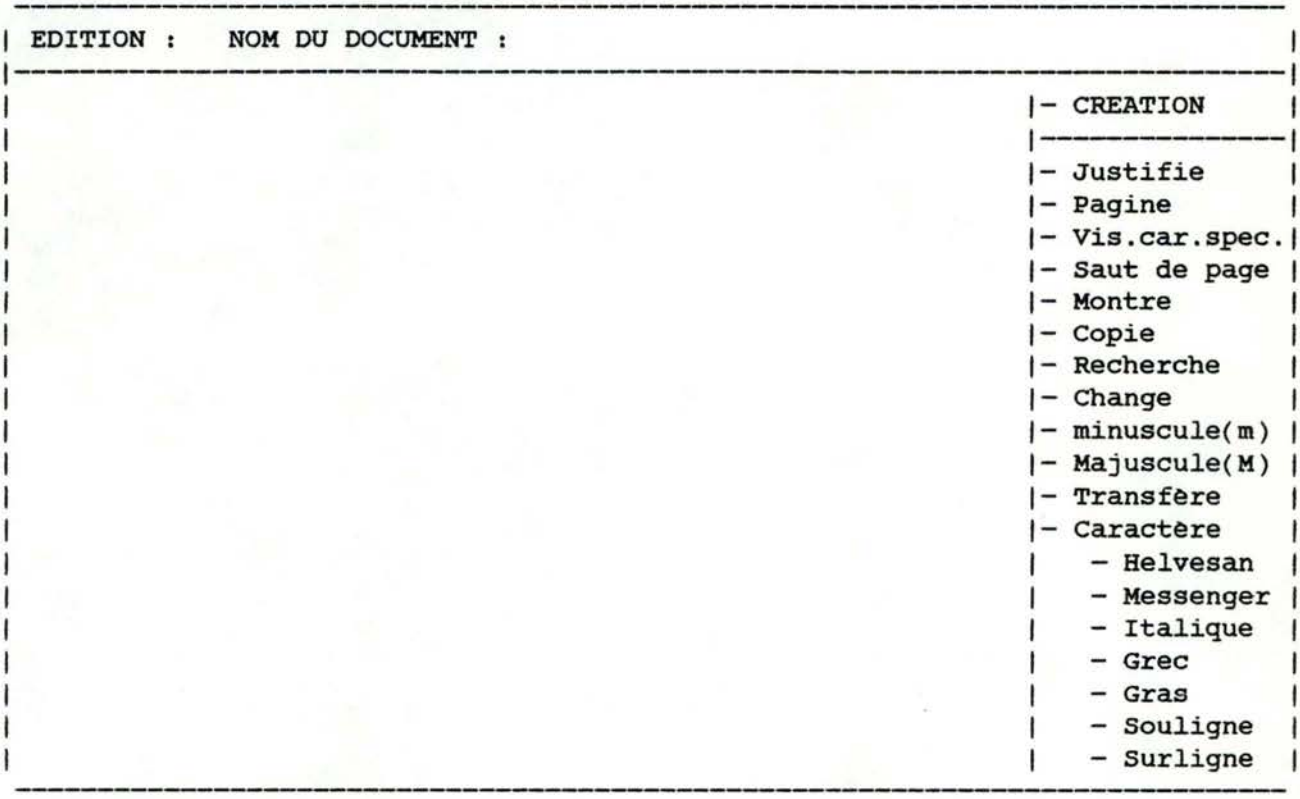
SYNONYME: Ecran2-q

DESCRIPTION:



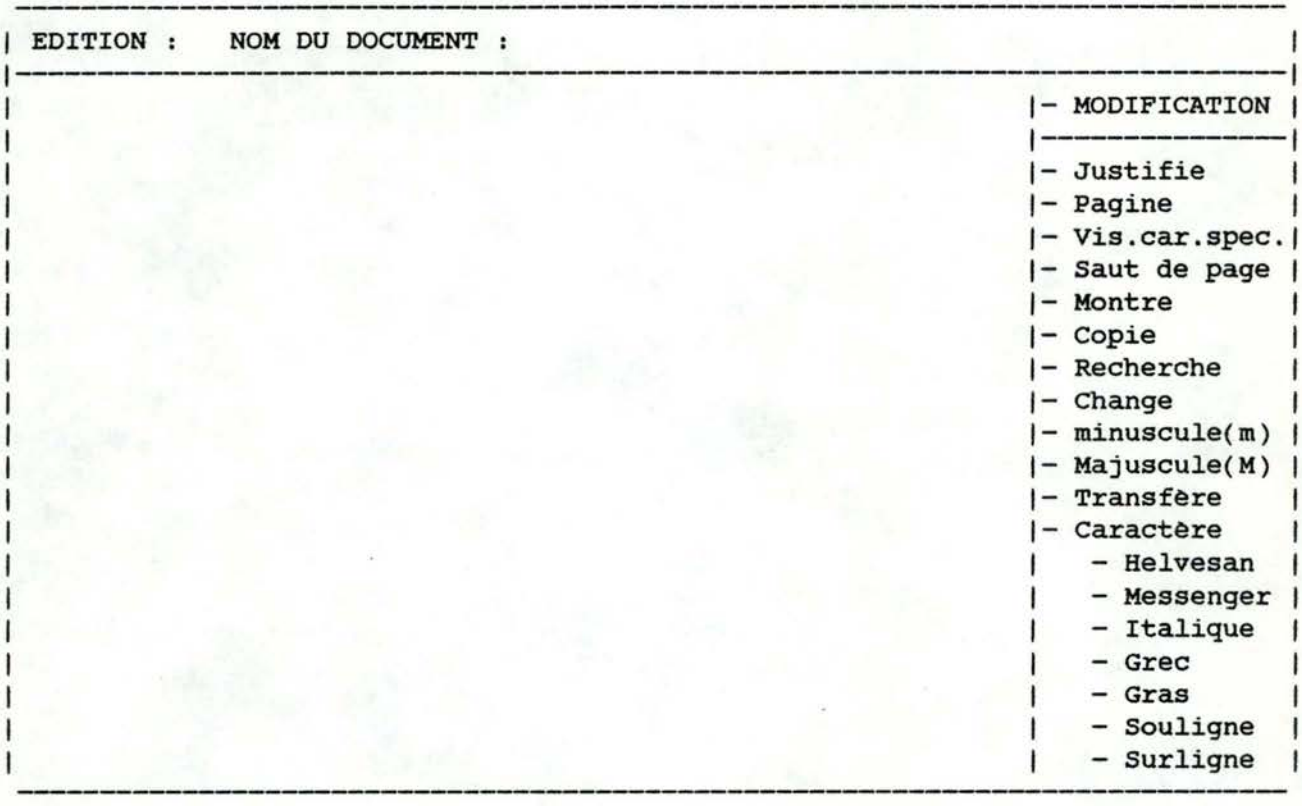
MESSAGE: Menu-caractère-2-c

DESCRIPTION:



MESSAGE: Menu-caractère-2-m

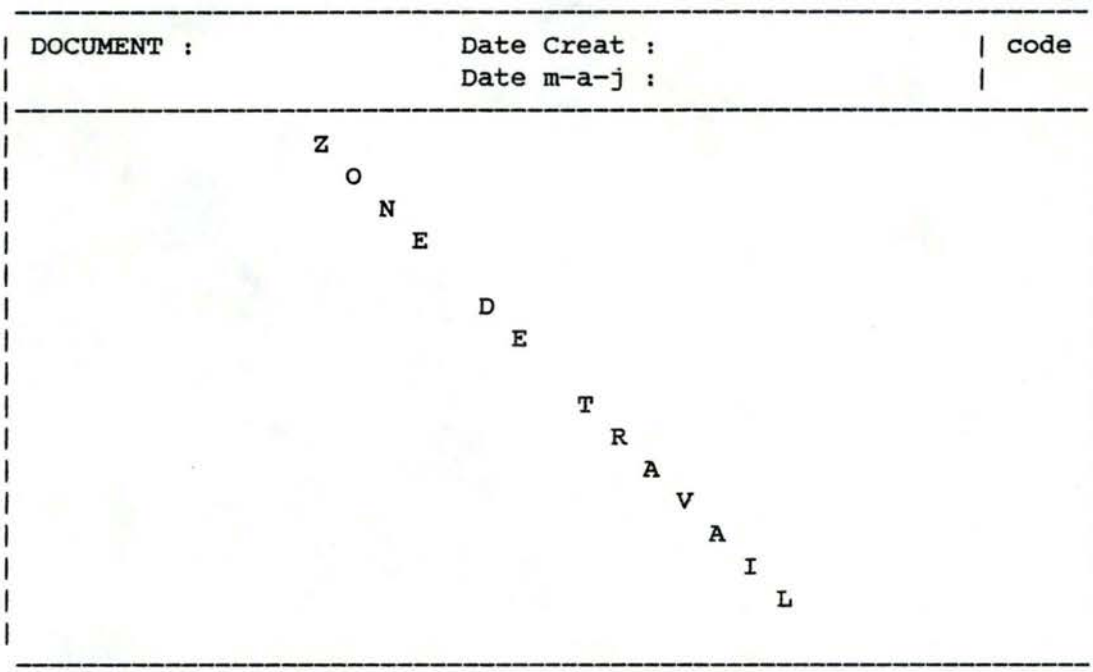
DESCRIPTION:



MESSAGE: Fenetre-document

DESCRIPTION:

Le document est divise en 2 zones : la zone d'identification et la zone de travail.



La zone d'identification comprend le nom du document, sa date de creation, sa date de dernière mise à jour ainsi que son code. La zone de travail comprend le document lui-meme.

Message: Choix-util-0

Description.

Ce message est le choix que fait l'utilisateur dans le menu initialisation.

Message: Choix-util-1

Description.

Ce message est le choix que l'utilisateur fait dans le menu-impr.

Message: Choix-util-2

Description.

Ce message est le choix que l'utilisateur fait lors de l'apparition du menu dans la phase d'édition.

Message: Choix-util-2-c

Description.

ce message est le choix que l'utilisateur fait lors de l'apparition du menu-creat dans la phase d'édition.

Message: Choix-util-2-m

Description.

Ce message est le choix que fait l'utilisateur lors de l'apparition du menu-mod dans la phase d'édition.

Message: Choix-util-2-*-c

Synonyme.

Choix-util-2-m-c

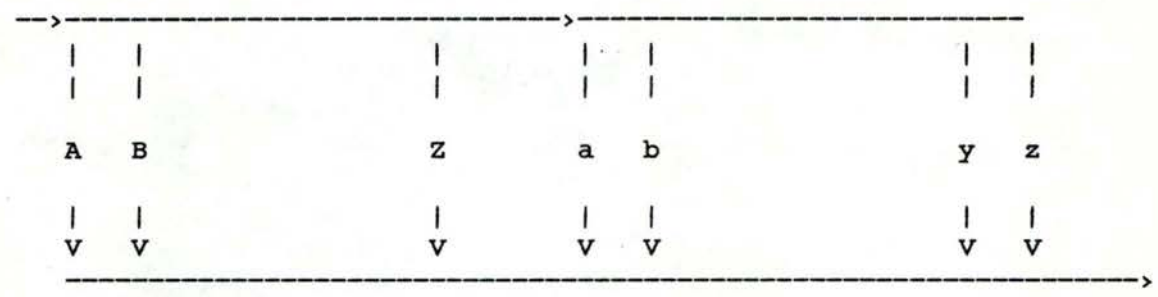
Choix-util-2-c-c

Description.

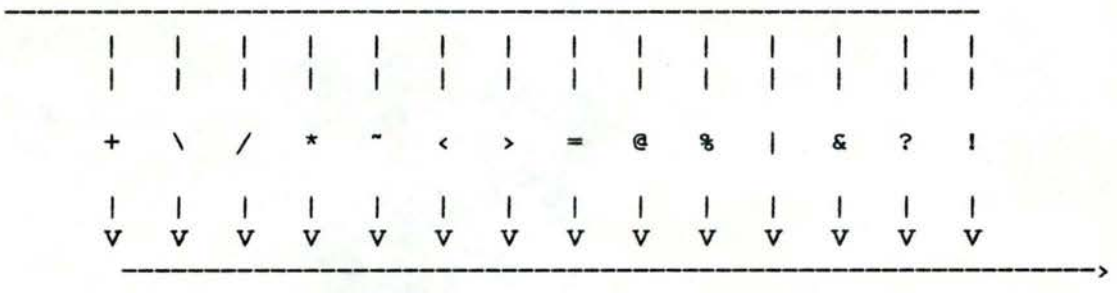
Ce message apparait lorsque l'utilisateur se trouve dans la primitive Caractère pour permettre à l'utilisateur de choisir quel style choisir.

4. Dictionnaire des données

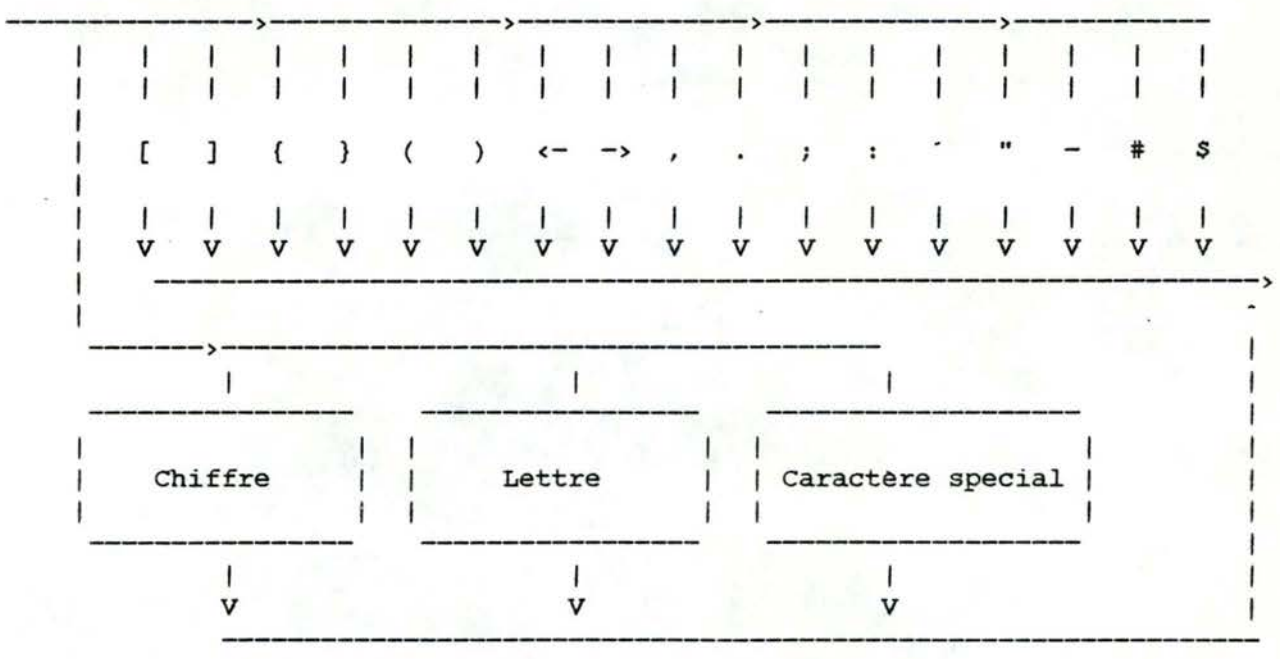
Lettre:



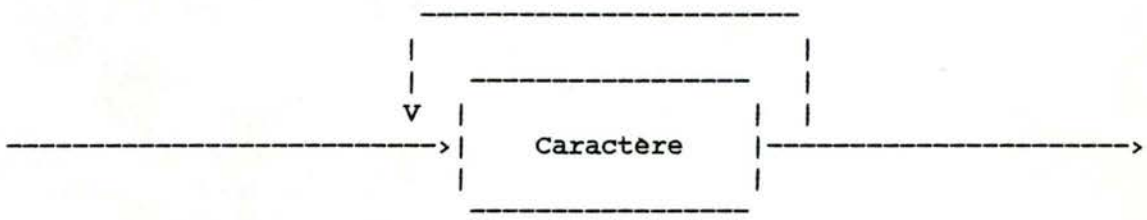
Caractère spécial :



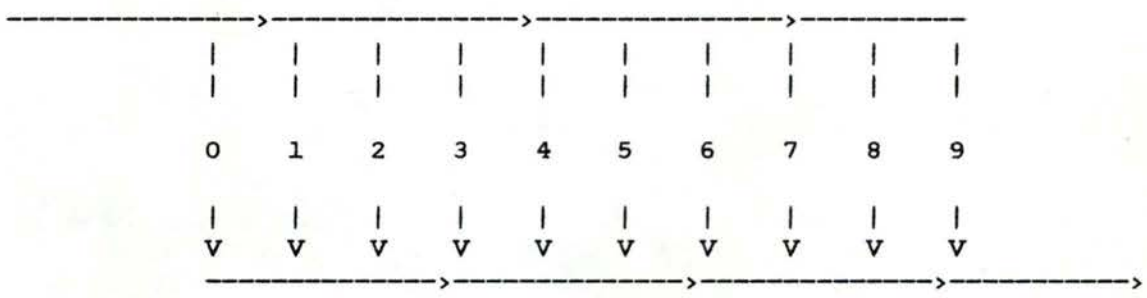
Caractère :



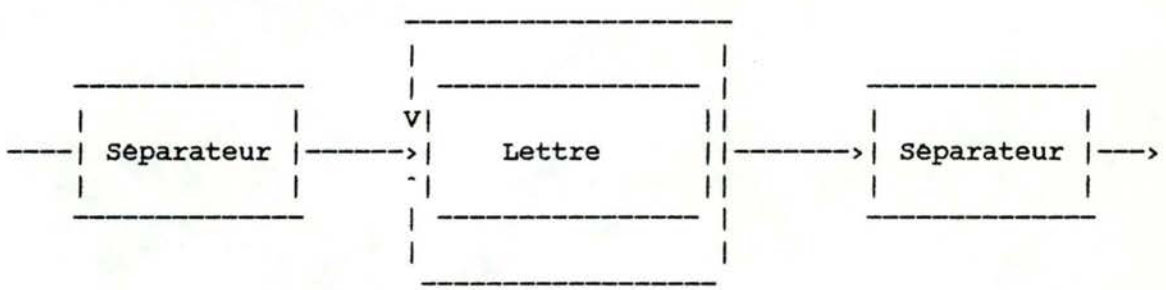
Chaîne de caractères :



Chiffre :

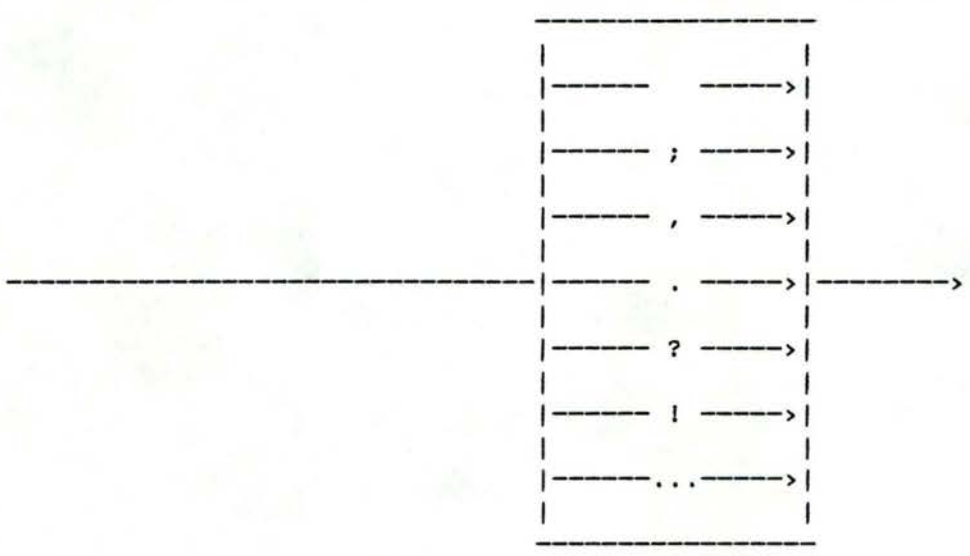


Mot :

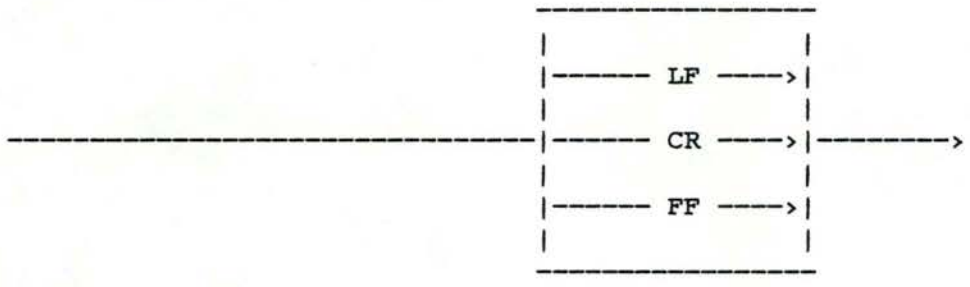


La caractéristique principale d'un mot est : Césure (coupure du mot). Les deux solutions possibles sont : couper le mot ou ne pas couper le mot.

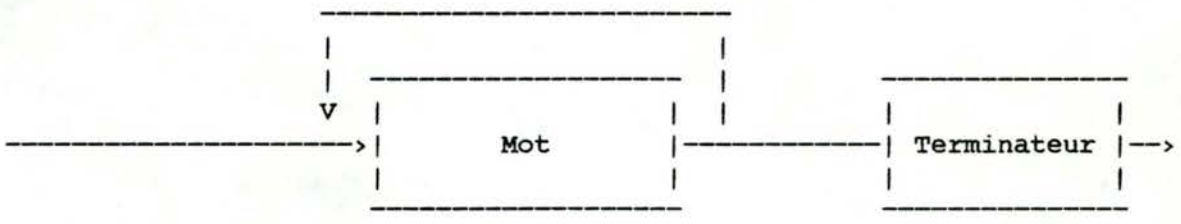
Séparateur :



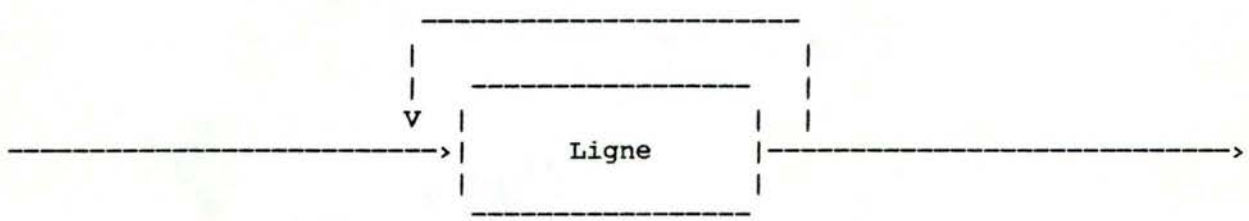
Terminateur :



Ligne :



Page :

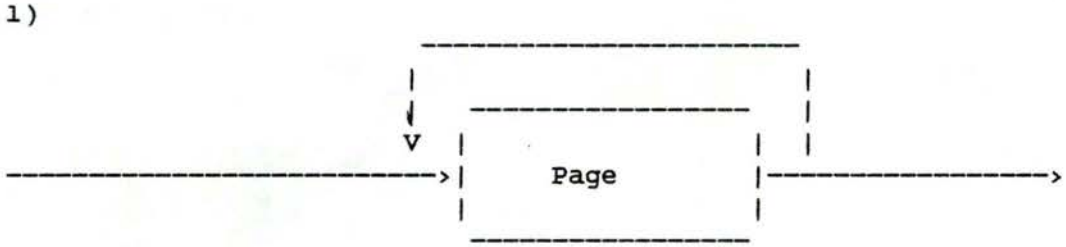


Une page est caractérisée par un numéro.

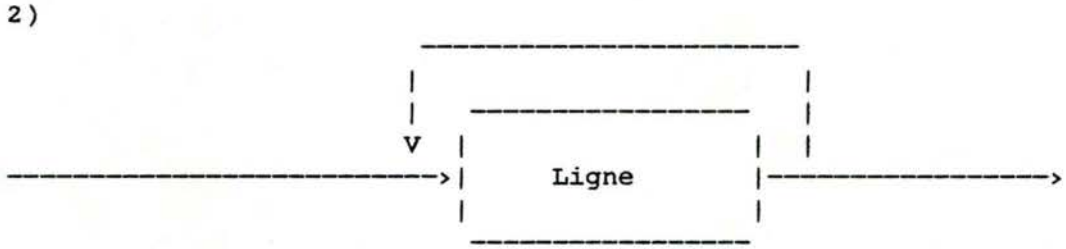
Texte :

Un synonyme de texte est document.
Un texte peut avoir trois définitions différentes

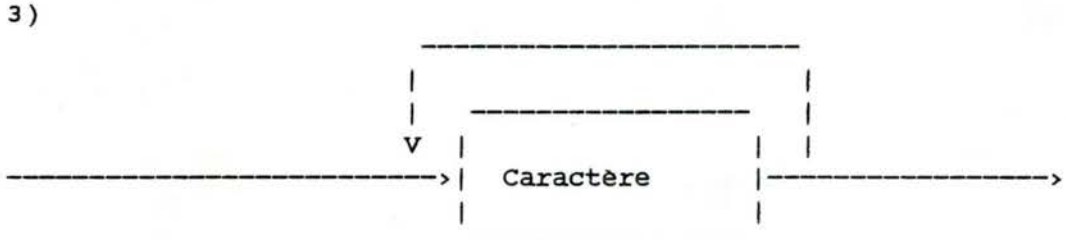
1)



2)



3)



Les caractéristiques d'un texte sont les suivantes :

- Nom : chaine de caractères de 14 caractères de long .

- Date de création du texte : date sous la forme (JJ MMM AA HH:mm).

- Date de mise-à-jour : date sous le forme (JJ MMM AA HH:mm).

- Marge gauche : nombre de caractères blancs à gauche d'un texte. La marge gauche doit être comprise entre 0 et 100 caractères blancs.

- Largeur de texte : nombre de caractères que comportera une ligne du texte. La largeur du texte doit être comprise entre 0 et 100 caractères.

- Interligne : espace qui est entre deux lignes écrites ou imprimées. L'interligne doit être compris entre 1 et 4. Il y a donc quatre interlignes possibles comme sur la plupart des machines à écrire.

- Marge haut : nombre de lignes blanches en haut d'un texte. La marge haut doit être comprise entre 0 et 66 lignes blanches.

- Longueur de page : nombre maximum de lignes de texte sur une page. La longueur de page doit être comprise entre 0 et 66 lignes.

- Tabulateur : dispositif permettant des signes en colonne, en tableau et de faire de l'indentation automatique. Il y a 5 tabulateurs permis au maximum. Chaque tabulateur(i) doit être compris entre le maximum(0,tabulateur(i-1)) et 100.

- Fonte : un jeu de caractères d'une certaine forme (style) et d'une certaine taille est appelé fonte ou police de caractère. Dans le cas de notre éditeur,les différentes fontes possibles sont : messenger, grec, italique, gras, helvesan.

- Justification à gauche : cadrage du texte sur une marge gauche. Les deux solutions possibles sont : justifié à gauche ou non justifié à gauche.

- Justification à gauche et à droite : cadrage du texte sur une marge gauche et sur une marge droite. Les deux solutions possibles sont : justifié à gauche et à droite ou non justifé à gauche et à droite.

- Vérification de la césure : possibilité de changer la césure des mots. Les deux possibilites sont : vérifier la coupure des mots ou ne pas vérifier la coupure des mots.

- Nombre maximum de blancs entre deux mots : détermine l'espacement maximum entre deux mots. Ce espace doit être supérieur ou égal à 1.

ECRAN: mémoire permettant l'affichage de fenêtres-document. Dimension

24 lignes de 200 caractères.

CURSOR: curseur de l'écran caractérisé par sa postion en y (YCURSOR) et sa position en x (XCURSOR).

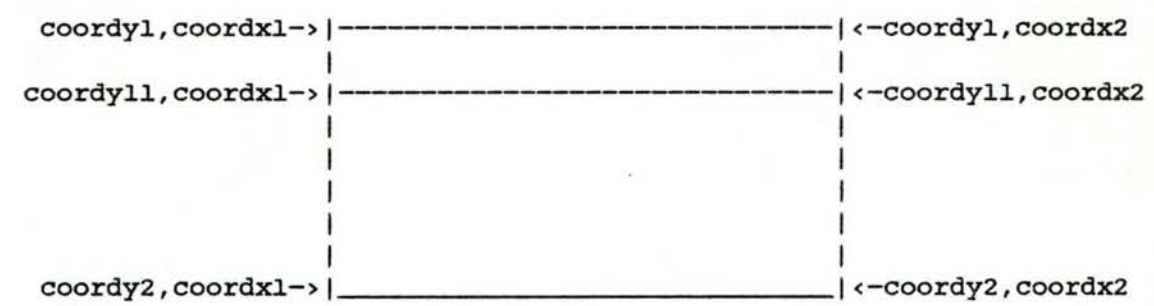
Fenêtre: zone de l'écran affichant différents renseignements. Il y a plusieurs sortes de fenêtre :

- fenêtre d'entête
- fenêtre de menu

Fenêtre d'entête: zone de l'écran comportant l'identification du traitement effectue ainsi que ses paramètres qui sont le type d'opérations dans le traitement et le nom du document sur lequel est effectuée l'opération.

Fenêtre de menu: zone l'écran comportant un menu des primitives que l'on peut executer dans le traitement effectué.

Fenêtre document: fenêtre de visualisation du document permettant de voir une partie du document plus ou moins grande selon la grandeur de celle-ci. A chaque fenêtre document correspond un descripteur de fenêtre. Chaque fenêtre de document comporte une zone d'identification et une zone de travail. La fenêtre document est caractérisée par des coordonnées. Ces coordonnées sont coordy1, coordy11, coordy2, coordx1,coordx2.



Ensuite, la largeur (larg_fen) et la longueur (lon_fen) peuvent être calculées.

Fenêtre d'identification: chaque fenêtre est identifiée par le nom du document,sa date de création,sa date de dernière modification et son code. Celui-ci varie selon la nature du document.

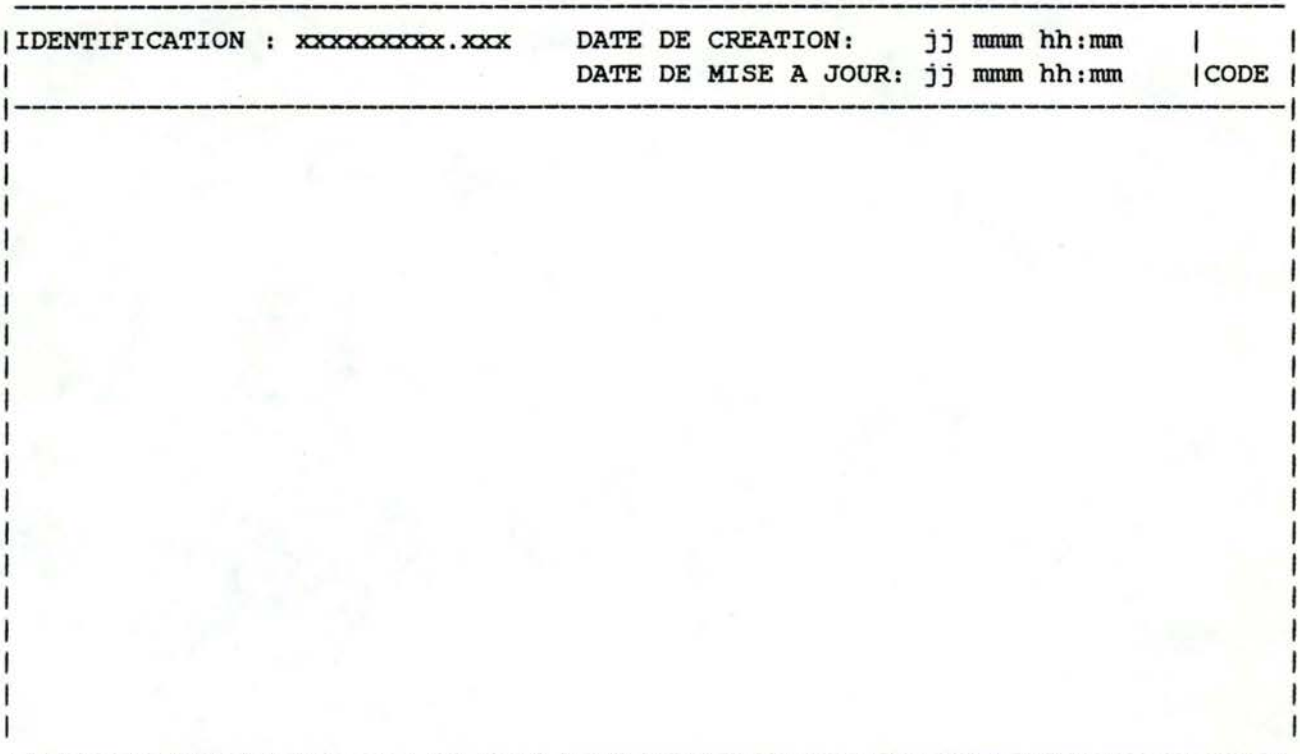
Zone de travail: c'est soit une zone graphique,soit une zone alphanumérique ou apparaitra un texte formate lu ou édité. Cette zone peut etre éditée par des primitives.

Texte formaté: Texte mis en page.

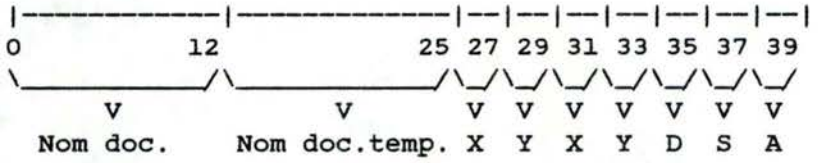
Texte au kilomètre : Texte contenant les indications de mises en pages.

Code: Zone alphabétique de trois caractères dont le contenu varie selon le type de document :

TXT----> texte.
FMT----> format.



Descripteur de fenêtre: zone en mémoire contenant l'identificateur de la fenêtre(nom du document),le nom du document temporaire,renseignements sur l'adresse du début de la fenêtre document,renseignements sur l'adresse de la fin de la fenêtre document et des flags(Destruction,Sauvetage,Activation).



DESCRTAB: table de descripteurs de fenêtre.

Menu : ensemble de possibilités de choix d'opérations à exécuter sur un document.

Fenêtre d'erreur: fenêtre contenant les messages d'erreur envoyés par le système en cas d'erreur de manipulation de l'utilisateur.

Souris : dispositif permettant de se positionner rapidement dans l'écran et comportant des boutons de sélection.

Chaîne recherchée : Chaîne de caractères qui est recherchée dans un texte.

Chaîne substitut : Chaîne de caractères qui doit être substituée à la chaîne recherchée.

Numèro de ligne courante : pointeur de ligne courante.

Numèro de ligne dèbut : pointeur dèlimitant le dèbut d'une portion de texte.

Numèro de ligne fin : pointeur dèlimitant la fin d'une portion de texte.

Numèro de ligne de transfert : pointeur de l'endroit où doit être transferèe une portion de texte dèlimitèe.

Numèro de caractère dèbut : pointeur de dèbut de portion de texte dans la ligne pointèe par Numèro de ligne dèbut.

Numèro de caractère fin: pointeur de fin de portion de texte dans la ligne pointèe par Numèro de ligne fin.

Marge gauche : nombre de caractères blancs à gauche d'un texte. La marge gauche doit être comprise entre 0 et 100 caractères blancs.

Largeur de texte : nombre de caractères que comportera une ligne du texte. La largeur du texte doit être comprise entre 0 et 100 caractères.

Interligne : espace qui est entre deux lignes écrites ou imprimées. L'interligne doit être compris entre 1 et 4. Il y a donc quatre interlignes possibles comme sur la plupart des machines à écrire.

Marge haut : nombre de lignes blanches en haut d'un texte. La marge haut doit être comprise entre 0 et 66 lignes blanches.

Longueur de page : nombre maximum de lignes de texte sur une page. La longueur de page doit être comprise entre 0 et 66 lignes.

Tabulateur : dispositif permettant des signes en colonne, en tableau et de faire de l'indentation automatique. Il y a 5 tabulateurs permis au maximum. Chaque tabulateur(i) doit être compris entre le maximum(0,tabulateur(i-1)) et 100.

Fonte : un jeu de caractères d'une certaine forme (style) et d'une certaine taille est appelè fonte ou police de caractère. Dans le cas de notre èditeur,les diffèrentes fontes possibles sont : messenger, grec, italique, gras, helvesan.

Justification à gauche : cadrage du texte sur une marge gauche. Les deux solutions possibles sont : justifiè à gauche ou non justifiè à gauche.

Justification à gauche et à droite : cadrage du texte sur une marge gauche et sur une marge droite. Les deux solutions possibles sont : justifiè à gauche et à droite ou non justifè à gauche et à droite.

Vèrification de la cèsure : possibilitè de changer la cèsure des mots. Les deux possibilites sont : vèrifier la coupure des mots ou ne pas vèrifier la coupure des mots.

Nombre maximum de blancs entre deux mots : détermine l'espacement maximum entre deux mots. Ce espace doit être supérieur ou égal à 1.

CADRATIN: pseudo-blanc utilisé afin de lier des caractères ou des mots qui ne peuvent être séparés dans la justification.

TIRET: pseudo-tiret ajouté par la justification lors de la césure des mots.

FF: marque de fin de page.

EOL: pseudo-retour de chariot ajouté par la justification.

CR: retour de chariot (fin de ligne).

SAJEX: pseudo-blanc ajouté par la justification.

LF : Saut de ligne.

ANNEXE B

PROGRAMMES

Annexe B: Programmes

1. Introduction.

Les programmes réalisés étant trop volumineux pour être placés dans cette partie du mémoire, nous avons laissé ceux-ci dans un partie séparée du mémoire.

Les programmes n'ont été imprimés qu'en deux exemplaires. Ces deux exemplaires sont disponibles chez Monsieur Van Bastelaer.

ANNEXE C

DISPOSITIF DE DEPLACEMENT DU CURSEUR : LA SOURIS

Annexe C: Dispositif de déplacement du curseur : la souris

La souris est un dispositif particulier de déplacement du curseur. Son aspect extérieur se présente généralement sous la forme d'un parallélipipède ou sous la forme d'une demi-sphère. Elle possède un ou plusieurs boutons de sélection.

Expliquons maintenant le mécanisme d'une souris de façon assez générale. Une souris se compose d'une bille en acier, qui, en roulant sur une surface plane, décompose son mouvement selon deux axes perpendiculaires. Ces deux axes perpendiculaires représentent l'abscisse et l'ordonnée du déplacement. Les deux signaux X et Y sont décomposé chacun en trains d'impulsions.

D'autres dispositifs semblables existent : le palpeur, le manche à balai, le crayon lumineux.

Le palpeur est un moyen similaire à la souris mais il se présente différemment : c'est une surface plane sur laquelle on se déplace par légère pression continue du doigt.

Le manche à balai se présente sous la forme d'un manche soudé sur une bille, il suffit de déplacer le manche pour déplacer le curseur.

Le crayon lumineux se présente sous la forme d'un crayon dont le rayonnement est perçu par l'écran lorsqu'on l'approche.

Parlons maintenant des applications possibles du dispositif souris. Comme première application, on peut imaginer aisément que la souris peut apporter une grande aide à l'encodage de schémas et de dessins. Disposant du dessin sur papier, on peut mémoriser ce dessin en suivant chaque ligne des contours avec la souris. Le programme mémorise le déplacement de la souris.

Dans une deuxième application, la souris peut permettre l'encodage dans le domaine médical de traitements à effectuer pour chaque patient. L'infirmière disposerait d'une feuille où sont notés les traitements. L'infirmière initialise pour l'ordinateur le début et la fin de la feuille. Ensuite partant du début , l'infirmière fait avancer la souris du point début jusqu'au point fin et chaque fois que la souris passe devant un traitement à effectuer, l'infirmière pousse sur le bouton de sélection ; suivant de déplacement effectué depuis le point début, l'ordinateur sait de quel traitement il s'agit.

Dans le cas de notre éditeur, la possibilité d'utiliser la souris faciliterait encore l'utilisation de l'outil réalisé. La souris permettrait d'effectuer des déplacements de curseur par un moyen différent du clavier. Ce nouveau dispositif simplifierait, pour l'utilisateur, les procédures de choix dans des menus et les procedures d'utilisation de certaines primitives d'action sur le texte où l'utilisateur doit délimiter une portion de texte.